ASEK-30

Reference Guide: Programming Using AASP DLLs

DESCRIPTION

This guide provides a reference for programming Allegro sensors using the Allegro Application Sample Programmer (AASP) dynamic-link libraries (DLLs).

Table of Contents

AASP COMMANDS

Creation and Communication

AASP

Initialize the base objects.

```
aasp.AASP();
```

GetCommunicationPortNames( )

Retrieves a list of port names for the vendor and product identifier, filtered by SetCommunicationPortFilters.

Parameter	Type	Description
return value	string[ ]	An array of communication port names

```
string[] port_names = aasp.GetCommunicationPortNames(
VendorID, ProductID);
```

GetCommunicationPortNames(int, int)

Retrieves a list of port names for the desired vendor and product identifier.

Parameter	Type	Description
VendorID	int	USB vendor identifier; 12824 (0x3218) is the Allegro vendor identifier
ProductID	int	USB product identifier: If all vendor products is desired, set to 1
return value	string[ ]	An array of communication port names

```
string[] port_names = aasp.GetCommunicationPortNames(
VendorID, ProductID);
```

SetCommunicationPortFilters(int, int, bool)

Sets the port filters for the desired vendor and product identifier.

Parameter	Type	Description
VendorID	int	USB vendor ID 12824 (0x3218) is the Allegro vendor identifier
ProductID	int	USB product ID: For all vendor products, set to 1
IncludeAllSerialPorts	bool	If all serial ports are to be included in the list (unfiltered by vendor and product identifiers), set to true
return value	string[ ]	An array of communication port names

```
string[] port_names = aasp.SetCommunicationPortNames(
VendorID, ProductID, true);
```

SetCommunicationPortFilters(int, int, bool)

Sets the port filters for the desired vendor and product identifier.

Parameter	Type	Description
FilterItems	PortFilterItem[ ]	An array of vendor and product identifiers
IncludeAllSerialPorts	bool	If all serial ports are to be included in the list (unfiltered by vendor and product identifiers), set to true
return value	string[ ]	An array of communication port names

```
AASP.PortFilterItem[] PortFilters = new AASP.
PortFilterItem[1];
PortFilters[0] = new AASP.PortFilterItem(0x3218, -1);
string[] port_names = aasp.SetCommunicationPortNames(
PortFilters, true);
```

GetCommunicationPort( )

Obtains the name of the port opened to the AASP.

Parameter	Type	Description
return value	string	The port name

```
string port_name = aasp.GetCommunicationPort();
```

OpenCommunicationPort(string)

Opens a port to an AASP.

Parameter	Type	Description
Port	string	The name of the serial or USB port to open
return value	bool	If the port was opened, the value is true

```
bool isOpened = OpenCommunicationPort(port);
```

CloseCommunicationPort( )

Closes the communication port.

```
aasp.CloseCommunicationPort();
```

IsActive( )

Checks to see if there is an active connection to the AASP.

Parameter	Type	Description
return value	bool	If there is an active connection to an AASP or ASEK, the value is true

```
bool isActive = aasp.IsActive();
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Programmer-Specific Commands

Version(out VersionInformation version_information)

Returns the version information of the connected programmer.

```
public class VersionInformation
{
        // The board (TED) number of the programmer
        public int Board { get; set; }

        // The major version number
        public int Major { get; set; }

        // The minor version number
        public int Minor { get; set; }

        // The bug version number
        public int Bug { get; set; }

        // The build number
        public int Build { get; set; }

        // The name of the programmer
        public string Name { get; set; }
        // The unique id of the programmer
        public string ID { get; set; }

        // The serial number of the programmer
        public string SerialNumber { get; set; }
};
```

Parameter	Type	Description
version_ information	VersionInformation	The version information from the programmer
return value	int	If the version call fails, returns the error; otherwise, returns kNOERROR(0)

```
int error = aasp.Version(out version_information);
```

GetProgrammerPorts(out int)

Obtains the number of ports contained in the programmer.

Parameter	Type	Description
ports	int	Returns the number of ports in the programmer
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetProgrammerPorts(out ports);
```

GetProgrammerResources(out int, out int, out int, out int, out int)

Obtains the number of resources supported by the programmer.

Parameter	Type	Description
ports	int	Returns the number of ports in the programmer
adcs	int	Returns the number of ADCs in the programmer
dacs	int	Returns the number of DACs in the programmer
gpios	int	Returns the number of GPIOs in the programmer
port_gpios	int	Returns the number of GPIOs associated with a port in the programmer
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetProgrammerResources(out ports,
out adcs, out dacs, out gpios, out port_gpios);
```

GetLastErrors( )

Obtains the list of errors.

Parameter	Type	Description
return value	int[ ]	An array of error codes

```
int[] errors = aasp.GetLastError(reset);
```

ClearLastErrors( )

Clears the list of errors.

```
aasp.ClearLastErrors();
```

Reset( )

Resets the programmer.

Parameter	Type	Description
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.Reset();
```

RunBootLoader(UInt32)

Starts the boot loader in the programmer.

Parameter	Type	Description
code	UInt32	Unlock code to start the bootloader
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.RunBootLoader(UInt32 code)
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Programmer Daughterboard Commands

Every daughterboard must have a PCA9555 at address 0x27 that contains the identifier (ID) of the daughterboard. The Allegro test equipment document (TED) number is typically used for the ID. Up to five more PCA9555s can be used to drive relays, setup for input, or other functions. Each of these nondaughterboard ID PCA9555s are a bank of input/output (I/O) bits [0:4].

GetDaughterboardID(out int)

Obtains the daughterboard identifier.

Parameter	Type	Description
id	int	The daughterboard identifier
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetDaughterboardID(out id);
```

SetDaughterboardBitsDirections(int, uint)

Sets the direction of the daughterboard bits.

Parameter	Type	Description
bank	int	The bit bank to write [0:4]
value	uint	The directions of the 16 bits in the bank (1 = output; 0 = input)
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
nt error = aasp.SetDaughterboardBitsDirections(bank,
value);
```

WriteDaughterboardOutputBits(int, uint)

Writes the values to the daughterboard output bits.

Parameter	Type	Description
bank	int	The bit bank to write [0:4]
value	uint	The value of the 16 bits in the bank that are set up for output
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDaughterboardOutputBits(bank,
value);
```

SetDaughterboardOutputBits(int, uint)

Sets the values of the output bits.

Parameter	Type	Description
bank	int	The bit bank to write [0:4]
mask	uint	The mask of the 16 bits in the bank that are to be set: 1 = set; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetDaughterboardOutputBits(bank,
mask);
```

ClearDaughterboardOutputBits(int, uint)

Clears the values of the daughterboard output bits.

Parameter	Type	Description
bank	int	The bit bank to write [0:4]
mask	uint	The mask of the 16 bits in the bank that are to be cleared: 1 = cleared; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ClearDaughterboardOutputBits(bank,
mask);
```

ToggleDaughterboardOutputBits(int, uint)

Toggles the daughterboard output bits.

Parameter	Type	Description
bank	int	The bit bank to write [0:4]
mask	uint	The mask of the 16 bits in the bank that are to be toggled: 1 = toggled; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ToggleDaughterboardOutputBits(bank,
mask);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ReadDaughterboardOutputBits(int, out uint)

Reads the daughterboard output bits and returns the values.

Parameter	Type	Description
bank	int	The bit bank to read [0:4]
value	uint	The value that was written of the 16 bits in the bank that are set for output
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDaughterboardOutputBits(bank,
out value);
```

ReadDaughterboardInputBits(int, out uint)

Reads the daughterboard input bits and returns the values.

Parameter	Type	Description
bank	int	The bit bank to read [0:4]
value	uint	The value that was read of the 16 bits in the bank that are set for input
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDaughterboardInputBits(bank, out
value);
```

Flags used for Daughterboard I2C Routines

Name	Type	Value	Description
kI2CDataMSBFirst	UInt16	0x0001	The msb of the data is the first byte read or written
kI2CAddressMSBFirst	UInt16	0x0100	The msb of the address is the first byte written
kI2CAddressSize	UInt16	0x0600	The size of the address sent to the I²C device, mask
kI2CAddressSize1	UInt16	0x0000	The size of the address sent to the I²C device is 1 byte
kI2CAddressSize2	UInt16	0x0200	The size of the address sent to the I²C device is 2 bytes
kI2CAddressSize3	UInt16	0x0400	The size of the address sent to the I²C device is 3 bytes
kI2CAddressSize4	UInt16	0x0600	The size of the address sent to the I²C device is 4 bytes

ReadDaughterboardByteI2C(int, long, int, out byte)

Reads a byte from a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
value	byte	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDaughterboardByteI2C(device_
address, address, flags, out value);
```

ReadDaughterboardWordI2C(int, long, int, out UInt16)

Reads a 16-bit word from a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
value	UInt16	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDaughterboardWordI2C(device_
address, address, flags, out value);
```

ReadDaughterboardLongI2C(int, long, int, out UInt32)

Reads a 32-bit word from a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
value	UInt32	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDaughterboardLongI2C(device_
address, address, flags, out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ReadDaughterboardByteArrayI2C(int, long, int, int, out byte[ ])

Reads a byte from a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
number_of_bytes	int	The number of bytes to read
value	byte[ ]	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDaughterboardByteArrayI2C(dev
ice_address, address, flags, number_of_bytes, out
values);
```

WriteDaughterboardByteI2C(int, long, int, byte)

Writes a byte to a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
value	byte	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDaughterboardByteI2C(device_
address, address, flags, value);
```

WriteDaughterboardWordI2C(int, long, int, UInt16)

Writes a 16-bit word to a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
value	UInt16	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDaughterboardWordI2C(device_
address, address, flags, value);
```

WriteDaughterboardLongI2C(int, long, int, UInt32)

Writes a 32-bit word to a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
value	UInt32	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDaughterboardLongI2C(device_
address, address, flags, value);
```

WriteDaughterboardByteArrayI2C(int, long, int, byte[ ])

Writes a byte array to a device on the daughterboard I²C bus.

Parameter	Type	Description
device_address	int	I²C bus address [0:127]
address	long	The register address
flags	int	The flags describing how the data and address are handled
values	byte[ ]	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDaughterboardByteArrayI2C(dev
ice_address, address, flags, values);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Programmer Features

These are the different programmer features that can be set or obtained from the programmer. Not all features are available on all programmers. If a feature is not available on the programmer, kUNKNOWNPARAMETERERROR or kNOTIMPLEMENTEDERROR is returned. If the value given to set a parameter is not within the range of valid values, kBADPARAMETERVALUEERROR is returned. When a feature is set, the hardware updates to reflect the new value.

Programmer Features

Name	Feature ID	Description
ProgrammerGPIOVoltage	0	GPIO voltage (float)
ProgrammerGPIOPull-upSelection	1	Indicates which pull-up is used on the GPIO bus (uint): 0 = off > 0 = hardware specific
ProgrammerI2CVoltage	2	I^2C bus voltage (float)
ProgrammerI2CPull-upSelection	3	Indicates which pull-up is used on the I^2C bus (uint): 0 = off > 0 = hardware specific
ProgrammerSPIVoltage	4	SPI bus voltage (float)
ProgrammerGPIOEnable	5	0 = false; otherwise, true
ProgrammerI2CEnable	6	0 = false; otherwise, true
ProgrammerSPIEnable	7	0 = false; otherwise, true
EVComparitor1Threshold	10000	Voltage threshold value for comparator 1 (float)
EVComparitor2Threshold	10001	Voltage threshold value for comparator 2 (float)
EVComparitor3Threshold	10002	Voltage threshold value for comparator 3 (float)
EVComparitor4Threshold	10003	Voltage threshold value for comparator 4 (float)
EVInputMultiplexor1	10004	Input selection for multiplexor 1 (uint)
EVInputMultiplexor2	10005	Input selection for multiplexor 2 (uint)

SetFeature(string, UInt32)

Sets the feature. The feature name is one of the names in the Programmer Features table.

Parameter	Type	Description
feature	string	The programmer feature to change
value	UInt32	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetFeature("ProgrammerGPIOPull-upSelection", 1);
```

SetFeature(UInt16, UInt32)

Sets the feature. The feature is the enumeration value.

Parameter	Type	Description
feature	UInt16	The programmer feature to change
value	UInt32	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetFeature((UInt16)AASP.ProgrammerFeature.ProgrammerGPIOPull-upSelection, 1);
```

GetFeature(string, out UInt32)

Obtains the feature. The feature name is one of the names in the Programmer Features table.

Parameter	Type	Description
feature	string	The programmer feature that has the value that is to be obtained
value	UInt32	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetFeature("ProgrammerGPIOPull-upSelection", out value);
```

GetFeature(UInt16, out UInt32)

Obtains the feature. The feature is the enumeration value.

Parameter	Type	Description
feature	UInt16	The programmer feature that has the value that is to be obtained
value	UInt32	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetFeature((UInt16)AASP.ProgrammerFeature.ProgrammerGPIOPull-upSelection, out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SetFloatFeature(string, double)

Sets the float feature of the AASP. The feature name is one of the names in the Programmer Features table.

Parameter	Type	Description
feature	string	The programmer feature to change
value	double	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetFloatFeature("ProgrammerGPIOVolt
age", value);
```

SetFloatFeature(UInt16, double)

Sets the float feature of the AASP. The feature is the enumeration value.

Parameter	Type	Description
feature	UInt16	The programmer feature to change
value	double	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetFloatFeature(feature, value);
```

GetFloatFeature(string, out double)

Obtains the float feature. The feature is the enumeration value.

Parameter	Type	Description
feature	string	The programmer feature that has the value that is to be obtained
value	double	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetFloatFeature("ProgrammerGPIOVolt
age", out value);
```

GetFloatFeature(UInt16, out double)

Obtains the float feature.

Parameter	Type	Description
feature	UInt16	The programmer feature that has the value that is to be obtained
value	double	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetFloatFeature(feature, out value);
```

Programmer Properties

All programmer features can be accessed through properties of the AASP class. This means that they can be read and written as if they were variables.

WARNING: These are the only actions in the AASP class that initiate an exception if they fail.

ProgrammerGPIOVoltage

The voltage that is used on the GPIO drivers and pull-ups.

```
aasp.ProgrammerGPIOVoltage = 5.0;
```

ProgrammerGPIOPull-upSelection

Selection of pull-up used on the GPIOs.

```
aasp.ProgrammerGPIOPull-upSelection = 1;
```

ProgrammerI2CVoltage

The voltage that is used on the I^2C drivers and pull-ups.

```
aasp.ProgrammerI2CVoltage = 3.3;
```

ProgrammerI2CPull-upSelection

The pull-up that is used on the I^2C bus.

```
aasp.ProgrammerI2CPull-upSelection = 2;
```

ProgrammerSPIVoltage

The voltage that is used on the SPI drivers.

```
aasp.ProgrammerSPIVoltage = 3.3;
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ADC Commands

ReadADC(UInt16, UInt16, UInt16, out double)

Reads the ADC, and returns the value.

Parameter	Type	Description
address	UInt16	The ADC to read
multiplexor	UInt16	The selection value for the multiplexor connected to the ADC
number_of_ samples	UInt16	The number of samples to average
value	double	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadADC(address, multiplexor, out value);
```

ReadADCRaw(UInt16, UInt16, UInt16, out UInt32)

Reads the ADC, and returns the raw value.

Parameter	Type	Description
address	UInt16	The ADC to read
multiplexor	UInt16	The selection value for the multiplexor connected to the ADC
value	UInt32	The value that was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadADCRaw(address, multiplexor, out value);
```

ReadADCPair(UInt16, UInt16, UInt16, UInt16, out double, out double)

Reads the ADC pair, and returns the values. If possible, the reads are synchronized.

Parameter	Type	Description
address0	UInt16	The first ADC to read
address1	UInt16	The second ADC to read
multiplexor0	UInt16	The selection value for the multiplexor connected to the first ADC
multiplexor1	UInt16	The selection value for the multiplexor connected to the second ADC
value0	double	The value that was read from the first ADC
value1	double	The value that was read from the second ADC
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadADCPair(address0, address1, multiplexor0, multiplexor1, out value0, out value1);
```

ReadADCPairRaw(UInt16, UInt16, UInt16, UInt16, out UInt32, out UInt32)

Reads the ADC pair, and returns the raw values. If possible, the reads are synchronized.

Parameter	Type	Description
address0	UInt16	The first ADC to read
address1	UInt16	The second ADC to read
multiplexor0	UInt16	The selection value for the multiplexor connected to the first ADC
multiplexor1	UInt16	The selection value for the multiplexor connected to the second ADC
value0	UInt32	The value that was read from the first ADC
value1	UInt32	The value that was read from the second ADC
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadADCPairRaw(address0, address1, multiplexor0, multiplexor1, out value0, out value1);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

DAC Commands

WriteDAC(UInt16, double)

Writes the value to a DAC.

Parameter	Type	Description
address	UInt16	The DAC to write
value	double	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDAC(address, value);
```

WriteDACRaw(UInt16, UInt32)

Writes the value to a DAC.

Parameter	Type	Description
address	UInt16	The DAC to write
value	UInt32	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDACRaw(address, value);
```

WriteDACPair(UInt16, UInt16, double, double)

Writes the value to a DAC pair.

Parameter	Type	Description
address0	UInt16	The first DAC to write
address1	UInt16	The second DAC to write
value0	double	The value to write to the first ADC
value1	double	The value to write to the second ADC
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDACPair(address0, address1,
value0, value1);
```

WriteDACPairRaw(UInt16, UInt16, UInt32, UInt32)

Writes the raw value to a DAC pair.

Parameter	Type	Description
address0	UInt16	The first DAC to write
address1	UInt16	The second DAC to write
value0	UInt32	The value to write to the first ADC
value1	UInt32	The value to write to the second ADC
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDACPairRaw(address0, address1,
value0, value1);
```

WriteSequenceDAC(UInt16, double, double[ ])

Writes a sequence of values to a DAC.

Parameter	Type	Description
address	UInt16	The DAC to write
rate	double	The rate of performance of the write operation to the DAC, in Hz
value	double	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDAC(address, rate, value);
```

WriteSequenceDACRaw(UInt16, double, UInt32[ ])

Writes a sequence of raw values to a DAC.

Parameter	Type	Description
address	UInt16	The DAC to write
rate	double	The rate of performance of the write operation to the DAC, in Hz
value	UInt32	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteDACRaw(address, rate, value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

WriteSequenceDACPair(UInt16, UInt16, double, double[ ], double[ ])

Writes a sequence of values to a DAC pair.

Parameter	Type	Description
address0	UInt16	The first DAC to write
address1	UInt16	The second DAC to write
rate	double	The rate of performance of the write operation to the DAC, in Hz
value0	double[ ]	The values to write to the first ADC
value1	double[ ]	The values to write to the second ADC
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteSequenceDACPair(address1,
address2, rate, values1, values2);
```

WriteSequenceDACRaw(UInt16, UInt16, double, UInt32[ ], UInt32[ ])

Writes a sequence of raw values to a DAC pair.

Parameter	Type	Description
address0	UInt16	The first DAC to write
address1	UInt16	The second DAC to write
rate	double	The rate of performance of the write operation to the DAC, in Hz
value0	UInt32[ ]	The raw values to write to the first ADC
value1	UInt32[ ]	The raw values to write to the second ADC
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteSequenceDACPairRaw(address1,
address2, rate, values1, values2);
```

ReadDAC(UInt16, out double)

Reads the DAC, and returns the value.

Parameter	Type	Description
address	UInt16	The DAC to read
value	double	The value the was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDAC(address, double value);
```

ReadDACRaw(UInt16, out UInt32)

Reads the DAC, and returns the raw value.

Parameter	Type	Description
address	UInt16	The DAC to read
value	UInt32	The raw value the was read
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDACRaw(address, out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Programmer GPIO

SetOutputBitsDirections(UInt32)

Sets the direction of the output bits (1 = output, 0 = input). There are a maximum of 32 GPIO bits; however, the actual number depends on the hardware of the programmer.

Parameter	Type	Description
value	UInt32	The directions of the GPIOs: 1 = output; 0 = input
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetOutputBitsDirections(value);
```

WriteOutputBits(UInt32)

Writes the values to the output bits.

Parameter	Type	Description
value	UInt32	The values of the GPIOs
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WriteOutputBits(value);
```

SetOutputBits(UInt32)

Sets the bits in the output to 1 that are set to 1 in the mask; does not set any bits that are set to 0 in the mask.

Parameter	Type	Description
mask	UInt32	The mask of the bits in the GPIOs that are to be set: 1 = set; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetOutputBits(mask);
```

ClearOutputBits(UInt32)

Sets the bits in the output to 0 that are set to 1 in the mask; does not clear any bits that are set to 0 in the mask.

Parameter	Type	Description
mask	UInt32	The mask of the bits in the GPIOs that are to be cleared: 1 = clear; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ClearOutputBits(mask);
```

ToggleOutputBits(UInt32)

Toggles the bits in the output that are set to 1 in the mask; does not toggle any bits that are set to 0 in the mask.

Parameter	Type	Description
mask	UInt32	The mask of the bits in the GPIOs that are to be toggled: 1 = toggle; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ToggleOutputBits(mask);
```

ReadOutputBits(out uint)

Reads the output bits, and returns the values.

Parameter	Type	Description
value	UInt32	The value that was written to the GPIOs that are set for output
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadOutputBits(out value);
```

ReadInputBits(out uint)

Reads the input bits, and returns the values.

Parameter	Type	Description
value	UInt32	The value that was read from the GPIOs that are set for input
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadInputBits(out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Capture Event Commands

The programmer can time-stamp various input events. When the capture starts, the timer resets to 0, and all timestamps are relative to this start time.

Capturable Events

Name	Bit	Description
kCOMPARATOR1RISINGEVENT	0	Output of comparator 1 rising
kCOMPARATOR1FALLINGEVENT	1	Output of comparator 1 falling
kCOMPARATOR2RISINGEVENT	2	Output of comparator 2 rising
kCOMPARATOR2FALLINGEVENT	3	Output of comparator 2 falling
kCOMPARATOR3RISINGEVENT	4	Output of comparator 3 rising
kCOMPARATOR3FALLINGEVENT	5	Output of comparator 3 falling
kCOMPARATOR4RISINGEVENT	6	Output of comparator 4 rising
kCOMPARATOR4FALLINGEVENT	7	Output of comparator 4 falling
kMANCHESTER1RISINGEVENT	8	Output of Manchester 1 rising
kMANCHESTER1FALLINGEVENT	9	Output of Manchester 1 falling
kMANCHESTER2RISINGEVENT	10	Output of Manchester 2 rising
kMANCHESTER2FALLINGEVENT	11	Output of Manchester 2 falling
kGPIO0RISINGEVENT	12	GPIO 0 rising
kGPIO0FALLINGEVENT	13	GPIO 0 falling
kGPIO1RISINGEVENT	14	GPIO 1 rising
kGPIO1FALLINGEVENT	15	GPIO 1 falling
kGPIO2RISINGEVENT	16	GPIO 2 rising
kGPIO2FALLINGEVENT	17	GPIO 2 falling
kGPIO3RISINGEVENT	18	GPIO 3 rising
kGPIO3FALLINGEVENT	19	GPIO 3 falling
kGPIO4RISINGEVENT	20	GPIO 4 rising
kGPIO4FALLINGEVENT	21	GPIO 4 falling
kGPIO5RISINGEVENT	22	GPIO 5 rising
kGPIO5FALLINGEVENT	23	GPIO 5 falling
kGPIO6RISINGEVENT	24	GPIO 6 rising
kGPIO6FALLINGEVENT	25	GPIO 6 falling
kGPIO7RISINGEVENT	26	GPIO 7 rising
kGPIO7FALLINGEVENT	27	GPIO 7 falling

CapturedEvent Structure

Public class captured event.

```
{
 // Which event was captured.
 public UInt32 Event;

 // When the event was captured in seconds from the
start of capture.
 public double When;
};
```

StartEventCapture(UInt32, UInt32, UInt32, UInt32)

Starts capturing events.

Parameter	Type	Description
capture_flags	UInt32	The flags to modify the capture (not used)
event_flags	UInt32	Bit mask of events to be captured
starting_flags	UInt32	Bit mask of events that trigger the capture to begin
stopping_flags	UInt32	Bit mask of events that trigger the capture to conclude
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.StartEventCapture(capture_flags,
event_flags, starting_flags, stopping_flags);
```

StopEventCapture( )

Stops capturing events.

Parameter	Type	Description
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.StopEventCapture();
```

ReadCapturedEvents(out CapturedEvent[ ], out bool)

Reads captured events.

Parameter	Type	Description
events	CapturedEvent[ ]	The array of captured events
more_events	bool	If there are more captured events in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadCapturedEvents(out events, out
more_events);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

VCC Port Commands

SetVccOn(int, double, bool)

Sets the VCC port to the desired voltage.

Parameter	Type	Description
port	int	The port
voltage	double	The voltage at which V_{CC} is to be set
unlock_device	bool	If the access codes are to be sent to the devices, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetVccOn(port, voltage, unlock_
device);
```

SetVccOff(int)

Performs a turn-off of the port.

Parameter	Type	Description
port	int	The port
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetVccOff(port);
```

GetVcc(int, out double)

Obtains the voltage on the V_{CC} line of the port.

Parameter	Type	Description
port	int	The port
voltage	double	The actual voltage of V_{CC}
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetVcc(port, out voltage);
```

GetIcc(int, out double)

Obtains the current being supplied to the port.

Parameter	Type	Description
port	int	The port
current	double	The actual current being supplied on V_{CC}, in mA
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetIcc(port, out current);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Port Features

The different port features that can be set or obtained from the port. Not all features are available on all ports or programmers. If a feature is not available on the port or programmer, kUNKNOWNPARAMETERERROR is returned. If the value given to set a feature is not within the range of valid values, kBADPARAMETERVALUEERROR is returned. When a feature is set, the hardware is updated to reflect the new value.

Name	Feature ID	Description
PortVoutPull-upSelection	0	Pull-up used on V_{OUT} (uint): 0 = off > 0 = hardware specific
PortVoutPull-upVoltage	1	Voltage used on the V_{OUT} pull-up (float)
PortInputHighThreshold	2	Input high-threshold voltage (float)
PortInputLowThreshold	3	Input low-threshold voltage (float)
PortGPIOVoltage	4	GPIO voltage (float)
PortGPIOPull-upSelection	5	Pull-up used on the GPIO bus (uint): 0 = off > 0 = hardware specific
PortI2CVoltage	6	I^2C bus voltage (float)
PortI2CPull-upSelection	7	Pull-up used on the I^2C bus (uint): 0 = off > 0 = hardware specific
PortSPIVoltage	8	SPI bus voltage (float)
PortVoutPull-downSelection	9	Pull-down used on V_{OUT} (uint): 0 = off > 0 = hardware specific
PortVoutCapacitorSelection	10	Capacitor used on V_{OUT}: 0 = off > 0 = hardware specific (uint)
PortVccBypassCapacitorEnable	11	Bypass capacitor used on V_{CC} (bool)
PortVccRisingSlewRate	12	Slew rate for the rising V_{CC}, in V/ms (float)
PortVccFallingSlewRate	13	Slew rate for the falling V_{CC}, in V/ms (float)
PortGPIOEnable	14	Enable port GPIOs (bool)
PortI2CEnable	15	Enable port I^2C bus (bool)
PortSPIEnable	16	Enable port SPI bus (bool)
PortVccOverCurrentLimitEnable	17	Enable port overcurrent limit for V_{CC} (bool)
PortVccOverCurrentLimit	18	Port overcurrent limit for V_{CC}, in A (float)
PortVoutOverCurrentLimitEnable	19	Enable port overcurrent limit for V_{OUT} (bool)
PortVoutOverCurrentLimit	20	Port overcurrent limit for V_{OUT}, in A (float)
SVClockFrequency	9000	Scan-vector clock frequency, in Hz (uint)
SVClockPolarity	9001	Clock active polarity for scan vector: false = active high; true = active low (bool)
SVClockPhase	9002	Clock phase for scan vector: false = data captured upon rising edge and output upon falling edge; true = data captured upon falling edge and output upon rising edge (bool)
SVOutputPolarity	9003	Output polarity: false = active low; true = active high (bool)
SVEnablePolarity	9004	Scan enable polarity: false = active low; true = active high (bool)
SVResetBit	9005	GPIO bit used for reset (uint)

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SetPortFeature(int, UInt16, UInt32)

Sets the feature of a port.

Parameter	Type	Description
port	int	The port where the feature is located
feature	UInt16	The port feature to change
value	UInt32	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetPortFeature(port, feature,
value);
```

GetPortFeature(int, UInt16, out UInt32)

Obtains the feature from a port.

Parameter	Type	Description
port	int	The port where the feature is located
feature	UInt16	The port feature value to be obtained
value	UInt32	The value that was read

```
int error = aasp.GetPortFeature(port, feature, out
value);
```

SetPortFloatFeature(int, UInt16, double)

Sets the float feature of a port.

Parameter	Type	Description
port	int	The port where the feature is located
feature	UInt16	The port feature to change
value	double	The value to write
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetPortFloatFeature(port, feature,
value);
```

GetPortFloatFeature(int, UInt16, out double)

Obtains the feature from a port.

Parameter	Type	Description
port	int	The port where the feature is located
feature	UInt16	The port feature value to be obtained
value	double	The value that was read

```
int error = aasp.GetPortFloatFeature(port, feature,
out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

GPIO Port Commands

SetPortBitsDirections(int, UInt32)

Sets the direction of the output bits on the port (1 = output, 0 = input). A maximum of 32 port GPIO bits is available; however, the actual number depends on the hardware of the programmer.

Parameter	Type	Description
port	int	The port where the GPIOs are located
value	UInt32	The directions of the GPIOs: 1 = output; 0 = input
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetPortBitsDirections(port, value);
```

WritePortOutputBits(int, UInt32)

Writes the values to the output bits.

Parameter	Type	Description
port	int	The port where the GPIOs are located
value	UInt32	The value of the GPIOs
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.WritePortOutputBits(port, value);
```

SetPortOutputBits(int, UInt32)

Sets the bits in the output to 1 that are set to 1 in the mask; does not set any bits that are set to 0 in the mask.

Parameter	Type	Description
port	int	The port where the GPIOs are located
mask	UInt32	The mask of the bits in the GPIOs that are to be set: 1 = set; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SetPortOutputBits(port, mask);
```

ClearPortOutputBits(int, UInt32)

Sets the bits in the output to 0 that are set to 1 in the mask; does not clear any bits that are set to 0 in the mask.

Parameter	Type	Description
port	int	The port where the GPIOs are located
mask	UInt32	The mask of the bits in the GPIOs that are to be cleared: 1 = clear; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ClearPortOutputBits(port, mask);
```

TogglePortOutputBits(int, UInt32)

Toggles the bits in the output that are set to 1 in the mask; does not toggle any bits that are set to 0 in the mask.

Parameter	Type	Description
port	int	The port where the GPIOs are located
mask	UInt32	The mask of the bits in the GPIOs that are to be toggled: 1 = toggle; 0 = no action
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.TogglePortOutputBits(port, mask);
```

ReadPortOutputBits(int, out UInt32)

Reads the output bits, and returns the values.

Parameter	Type	Description
port	int	The port where the GPIOs are located
values	UInt32	The values read from the GPIOs that are set for input
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadPortOutputBits(port, out values);
```

ReadPortInputBits(int, out UInt32)

Reads the input bits, and returns the values.

Parameter	Type	Description
port	int	The port where the GPIOs are located
values	UInt32	The values written to the GPIOs that are set for output
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadPortInputBits(port, out values);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Creation, Destruction, and Listing

CreateDevice(int, out int)

Creates a device.

Parameter	Type	Description
port	int	The port where the device is to be created
device	int	The identifier of the device that was created
return value	int	• kINVALIDPORTERROR: The programmer does not have the specified port • kTOOMANYDEVICESERROR: Addition of the specified device would exceed the capability of the programmer • kNOERROR(0): The above conditions do not apply

```
int error = aasp.CreateDevice(port, out device);
```

DestroyDevice(int)

Deletes the device from the AASP.

Parameter	Type	Description
device	int	The device identifier value
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table • kNOERROR(0): The above error did not occur

```
int error = aasp.DestroyDevice(device);
```

GetDevices(out int[ ])

Obtains the list of devices that are active.

Parameter	Type	Description
devices	int[ ]	The device identifier values
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.GetDevices(out devices);
```

GetDevicePort(int, out int)

Obtains the port of an active device.

Parameter	Type	Description
device	int	The device identifier value
port		The port where the device is connected
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table • kNOERROR(0): The above error did not occur

```
int error = aasp.GetDevicePort(int device_id, out int port)
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Parameter Values and Commands

The device is a construct that contains the communication parameters needed to communicate with a physical device. The parameter commands are used to change those parameters. Because a device might need to communicate with different protocols at the same time, storage is allocated for all protocols. Not all programmers implement all parameters.

Device Parameter Values

Name	Value	Description
DeviceProtocol	0	Protocol used by the device (uint): 0 = Protocol not used 1 = SPI 2 = I^2C 3 = Manchester 4 = Pulses
DeviceDelayAfterEepromWrite	1	Delay between EEPROM writes, in seconds (float)
DeviceOutputType	2	Output of the device (uint): 0 = Output not used 1 = Analog 2 = PWM 3 = SENT 4 = Other
DeviceDelayAfterPowerOn	3	Delay before commands at power-up, in seconds (float)
DeviceNeedPowerOnUnlock	4	Unlock required at power-up (bool)
DeviceUnlockAddresses	5	Addresses of the unlock register (uint)
DeviceUnlockCodes	6	Unlock code (uint[ ])
DeviceUnlockCodesWidth	7	Width of unlock code (uint)
DeviceNumberOfIndirectMemories	8	Device number of indirect memories (uint)
DeviceHasCommunicationEnable	9	Device has communication enabled (bool)
DeviceCommunicationEnableAddress	10	Device communication-enable address (uint)
DeviceCommunicationEnableMask	11	Device communication-enable mask (uint)
DeviceCommunicationEnableActiveState	12	Device communication-enable active state (bool)
DeviceCommunicationEnableInAccessCode	13	Device communication-enable is the lsb of the access code (bool)
DeviceProgramPulsesRequired	14	Device requires programming pulses (uint)
DeviceProgramPulseVoltage	15	Voltage level for program pulses, in V (float)
DeviceProgramPulseWidth	16	Width in seconds of program pulses (float)
DeviceProgramPulseDelay	17	Width in seconds of delay between program pulses (float)
DeviceProgramPulseOutputType	18	Location of application of program pulse (uint): 0 = Program pulses not applied 1 = Program pulses on V_{CC} 2 = Program pulses on V_{OUT} 3 = Program pulses on other
DevicePrologueWriteAddresses	19	Location of write during the prologue (uint[ ])
DevicePrologueWriteValues	20	Content for write during the prologue (uint[ ])
DevicePrologueReadAddresses	21	Location of read during the prologue (uint[ ])
DevicePrologueReadValues	22	Content for read during the prologue (uint[ ])
DevicePrologueReadMasks	23	Content to mask from the read during the prologue (uint[ ])
DeviceEpilogueWriteAddresses	24	Location of write during the epilogue (uint[ ])
DeviceEpilogueWriteValues	25	Content of write during the epilogue (uint[ ])
DeviceLowestEEPROMAddress	26	Lowest EEPROM address (uint)

Continued on next page...

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Parameter Values ... *continued from previous page*

Name	Value	Description
DeviceHighestEEPROMAddress	27	Highest EEPROM address (uint)
DeviceName	28	Device name (string)
DeviceInformation	29	Device information (string)
DeviceDisableEEPROMWrites	30	Disable writing to the device EEPROM (uint)
DeviceOutputReadAddress	31	Location to read the digital output (uint)
DeviceOutputIsSigned	32	Sign of the digital output (bool)
DeviceOutputReadMask	33	Bits to read for the digital output (uint)
DeviceIndirectReadAddressRegister	34	Indirect read address (uint)
DeviceIndirectReadAddressMask	35	Reads address mask (uint)
DeviceIndirectReadDataRegister	36	Read data register (uint)
DeviceIndirectReadDataWidth	37	Width of the data to be read (uint)
DeviceIndirectReadDataDirection	38	If the msb is the lowest address, the value is true (bool)
DeviceIndirectReadControlRegister	39	Read-data control register (uint)
DeviceIndirectReadCommand	40	Read-data command (uint[ ])
DeviceIndirectReadStatusRegister	41	Read-data status register (uint)
DeviceIndirectReadStatusMask	42	Read-data status mask (uint[ ])
DeviceIndirectReadStatus	43	Read-data status (uint[ ])
DeviceIndirectWriteAddressRegister	44	Indirect-write address (uint)
DeviceIndirectWriteAddressMask	45	Write-address mask (uint)
DeviceIndirectWriteDataRegister	46	Write-data register (uint)
DeviceIndirectWriteDataWidth	47	Width of the data to be written (uint)
DeviceIndirectWriteDataDirection	48	If the msb is the lowest address, the value is true (bool)
DeviceIndirectWriteControlRegister	49	Write-data control register (uint)
DeviceIndirectWriteCommand	50	Write-data command (uint[ ])
DeviceIndirectWriteStatusRegister	51	Write-data status register (uint)
DeviceIndirectWriteStatusMask	52	Write-data status mask (uint[ ])
DeviceIndirectWriteStatus	53	Write-data status (uint[ ])
DeviceIndirectMemoryWidth	54	Width of the indirect memory to be read or written (uint)
DeviceDirectMemoryWidth	55	Width of the data register to be directly read or written (uint)
DeviceProgramPulseRisingSlewRate	56	Slew rate of the rising program pulse, in V/ms (float)
DeviceProgramPulseFallingSlewRate	57	Slew rate of the falling program pulse, in V/ms (float)

I2C Parameter Values

Name	Value	Description
I2CClockFrequency	1000	I^2C clock frequency, in Hz (uint)
I2CAddress	1001	I^2C address of the device (uint)
I2CRegisterWidth	1002	Size of the register read or write (uint)
I2CByteAddressing	1003	Indicates if the device has byte addressing (uint)
I2CHasCRC	1004	Indicates if the device has a cyclic redundancy check (CRC) (bool)

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Manchester Parameter Values

Name	Value	Description
ManchesterSyncSize	2000	Size of the initial synchronization, in bits (uint)
ManchesterCommandSize	2001	Size of the command, in bits (uint)
ManchesterIDSize	2002	Size of the identifier, in bits (uint)
ManchesterFieldSelectSize	2003	Size of the field select, in bits (uint)
ManchesterAddressSize	2004	Size of the address, in bits (uint)
ManchesterReadDataSize	2005	Size of the read data, in bits (uint)
ManchesterWriteDataSize	2006	Size of the write data, in bits (uint)
ManchesterStatusSize	2007	Size of the status in a read response, in bits (uint)
ManchesterCRCSize	2008	Size of the CRC, in bits (uint)
ManchesterCommandOutput	2009	Location where the Manchester pulses are applied (uint): 0 = Not applied 1 = V_{CC} 2 = V_{OUT}, push-pull 3 = V_{OUT}, open-drain; requires a pull-up 4 = Other
ManchesterCommandEnableType	2010	Method required to enable communication with the device (uint): 0 = Not specified 1 = Raise V_{CC} 2 = Overdrive 3 = PWM function pulse 4 = SENT function pulse 5 = Trigger SENT and function pulse 6 = ASENT function pulse 7 = SSENT short function pulse 8 = SSENT long function pulse 12 = Raise V_{CC} and overdrive 13 = Raise V_{CC} and PWM function pulse 14 = Raise V_{CC} and SENT function pulse 15 = Raise V_{CC} and trigger SENT and function pulse 16 = Raise V_{CC} and ASENT function pulse 17 = Raise V_{CC} and SSENT short function pulse 18 = Raise V_{CC} and SSENT long function pulse
ManchesterDelayAfterCommandEnable	2011	Delay before first command after raising V_{CC} to command-enable voltage, in seconds (float)
ManchesterCommandEnableVccVoltage	2012	Voltage level to which V_{CC} is raised when performing a command-enable operation (float)
ManchesterCommandEnableVOutLevel	2013	Logical state to which V_{OUT} is raised when performing a command-enable operation (bool)
ManchesterHandlesMultipleCommands	2014	If set to true, multiple commands can be sent without use of program-enable cycling (bool)
ManchesterHasWriteResponse	2015	If the device has a write response, the value is true; only useful for EEPROM writes (bool)
ManchesterWriteResponseLevel	2016	Logical state to which the output transitions for the write response: 0 = high to low; 1 = low to high (bool)
ManchesterBitRate	2017	Bit rate, in bits per second (uint)
ManchesterSlewRate	2018	Slew rate, in V/ms (float)
ManchesterHighLevelVoltage	2019	Manchester high voltage (float)
ManchesterLowLevelVoltage	2020	Manchester low voltage (float)
ManchesterTriggerWidth	2021	Width of trigger, in seconds (float)
ManchesterDelayWidth	2022	Delay after trigger, in seconds (float)

Continued on next page...

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Manchester Parameter Values ... *continued from previous page*

Name	Value	Description
ManchesterAuxPulseWidth	2023	Width of auxiliary (aux) pulse, in seconds (float)
ManchesterHighDelayWidth	2024	Width of high-voltage delay, in seconds (float)
ManchesterLowDelayWidth	2025	Width of low-voltage delay, in seconds (float)
ManchesterBitTimesBeforeCommand	2026	Number of bit times before the command (uint)
ManchesterBitTimesAfterCommand	2027	Number of bit times after the command (uint)
ManchesterDelayAfterRead	2028	Delay after read, before next command, in seconds (float)
ManchesterDelayAfterWrite	2029	Delay after write, before next command, in seconds (float)
ManchesterCommandRetries	2031	Number of command retries: If a read fails or a write does not receive a write response, the command is retried (uint)
ManchesterNeedPull-upForRead	2032	If the device needs a pull-up during a read event, set to true; used when the device typically has a push-pull driver, except when responding to commands; has an open-drain driver (bool)
ManchesterResponseInput	2033	Location where the Manchester input is received (uint): 0 = None 1 = V_{OUT} 2 = Analog input 3 = Digital input 4 = I_{CC} 5 = Other
ManchesterID	2100	Manchester ID (uint)
ManchesterFieldSelect	2101	Field select value (uint)

SPI Parameter Values

Name	Value	Description
SPIClockFrequency	3000	SPI clock frequency, in Hz (uint)
SPIMSBFirst	3001	SPI msb is first when true; otherwise, lsb is first (bool)
SPIMode	3002	SPI mode (0 through 3) (uint)
SPISaferSPI	3003	If the device is using SafeSPI, the value is true (bool)
SPITransferSize	3004	Width of each transfer, in number of bits (uint)
SPIWriteCommand	3005	Bit pattern of write command (uint)
SPIReadCommand	3006	Bit patterns of read command (uint)
SPICommandFieldSize	3007	Width of command (uint)
SPICommandFieldShift	3008	Command shift (uint)
SPIAddressFieldSize	3009	Width of address (uint)
SPIAddressFieldShift	3010	Address shift (uint)
SPIDataFieldSize	3011	Width of data (uint)
SPIDataFieldShift	3012	Data shift (uint)
SPICRCFieldSize	3013	Width of CRC (uint)

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SENT Parameter Values

Name	Value	Description
SENTNumberOfNibbles	5000	Number of nibbles in SENT message (uint)
SENTTickTime	5001	Tick time, in seconds (float)
SENTIncludeSCNInCRC	5002	Include the status and control nibble in the CRC (bool)
SENTType	5003	Type of SENT (uint): 0 = Streaming SENT 1 = Triggered SENT 2 = Addressable SENT 3 = Sequential SENT
SENTSensorID	5004	Sensor identifier (uint)
SENTMaxSensorID	5005	Maximum sensor identifier (uint)
SENTTriggerWidth	5006	Width of TSENT trigger, in ticks (uint)
SENTAuxPulseWidth	5007	Width of ASENT and SSENT F_AUX pulse, in ticks (uint)
SENTAddressPulseWidth	5008	Width of ASENT incremental-address (IncAdr) pulse, in ticks; does not include the five-tick low period (uint)
SENTSlowSerialDataMode	5009	The slow serial data mode (uint): 0 = Standard 1 = Extended
SENTSyncPulseWidth	5010	Width of SSENT F_SYNC pulse, in ticks (uint)
SENTOutputPulseWidth	5011	Width of ASENT and SSENT F_OUTPUT pulses, in ticks (uint)
SENTSamplePulseWidth	5012	Width of ASENT and SSENT F_SAMPLE pulses, in ticks (uint)

Pulse Parameter Values

Name	Value	Description
PulseOutput	6000	Set locations where pulses are applied and responses are received (uint): 0 = Pulses not applied, responses not received 1 = Applied to V_{CC}, with response on V_{OUT} 2 = Applied to V_{CC}, with response on I_{CC} 3 = Applied to V_{OUT}, with response on V_{OUT} 4 = Other
PulseOnWidth	6001	Set width of pulse-on operation, in seconds (float)
PulseOffWidth	6002	Set width of pulse-off operation, in seconds (float)
PulseVoltageLevels	6003	Voltage levels (float[ ])
PulseRisingSlewRate	6004	Slew rate of the rising pulses, in V/ms (float)
PulseFallingSlewRate	6005	Slew rate of the falling pulses, in V/ms (float)

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SetParameter(int, UInt16, UInt32)

Sets the parameter of a device.

Parameter	Type	Description
device	int	The device that has the parameter that is to be changed
parameter	UInt16	The parameter to change
value	UInt32	The value to write
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kBADPARAMETERVALUEERROR: The parameter value is not within the range of permitted values • kNOERROR(0): The above errors did not occur

```
int error = aasp.SetParameter(device, parameter,
value);
```

SetFloatParameter(int, UInt16, double)

Sets the float parameter of a device.

Parameter	Type	Description
device	int	The device that has the parameter that is to be changed
parameter	UInt16	The parameter to change
value	double	The value to write
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kBADPARAMETERVALUEERROR: The parameter value is not within the range of permitted values • kNOERROR(0): The above errors did not occur

```
int error = aasp.SetFloatParameter(device, parameter,
value);
```

GetParameter(int, UInt16, out UInt32)

Obtains the parameter from a device.

Parameter	Type	Description
device	int	The device that has the parameter value that is to be obtained
parameter	UInt16	The parameter to obtain
value	UInt32	The value that was read
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kNOERROR(0): The above errors did not occur

```
int error = aasp.GetParameter(device, parameter, out
value);
```

GetFloatParameter(int, UInt16, out)

Obtains the parameter from a device.

Parameter	Type	Description
device	int	The device that has the parameter value that is to be obtained
parameter	UInt16	The parameter to obtained
value	double	The value that was read
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kNOERROR(0): The above errors did not occur

```
int error = aasp.GetFloatParameter(device, parameter,
out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SetStringParameter(int, UInt16, string)

Sets the parameter of a device.

Parameter	Type	Description
device	int	The device that has the parameter that is to be changed
parameter	UInt16	The parameter to change
value	string	The value to write
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kBADPARAMETERVALUEERROR: The parameter value is not within the range of permitted values • kNOERROR(0): The above errors did not occur

```
int error = aasp.SetStringParameter(device,
parameter, value);
```

SetArrayParameter(int, UInt16, UInt32[ ])

Sets the array parameter of a device.

Parameter	Type	Description
device	int	The device that has the parameter that is to be changed
parameter	UInt16	The parameter to change
value	UInt32[ ]	The value to write
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kBADPARAMETERVALUEERROR: The parameter value is not within the range of permitted values • kNOERROR(0): The above errors did not occur

```
int error = aasp.SetArrayParameter(device, parameter,
value);
```

GetStringParameter(int, UInt16, out string)

Obtains the parameter from a device.

Parameter	Type	Description
device	int	The device that has the parameter value that is to be obtained
parameter	UInt16	The parameter to obtain
value	string	The value that was read
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kNOERROR(0): The above errors did not occur

```
int error = aasp.GetStringParameter(device,
parameter, out value);
```

GetArrayParameter(int, UInt16, out UInt32[ ])

Obtains the array parameter from a device.

Parameter	Type	Description
device	int	The device that has the parameter value that is to be obtained
parameter	UInt16	The parameter to obtain
value	UInt32[ ]	The value that was read
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kNOERROR(0): The above errors did not occur

```
int error = aasp.GetArrayParameter(device, parameter,
out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SetFloatArrayParameter(int, UInt16, double[ ])

Sets the float array parameter of a device.

Parameter	Type	Description
device	int	The device that has the parameter that is to be changed
parameter	UInt16	The parameter to change
value	double[ ]	The value to write
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kBADPARAMETERVALUEERROR: The parameter value is not within the range of permitted values • kNOERROR(0): The above errors did not occur

```
int error = aasp.SetFloatArrayParameter(device,
parameter, value);
```

GetFloatArrayParameter(int, UInt16, out double[ ])

Obtains the float array parameter from a device.

Parameter	Type	Description
device	int	The device that has the parameter value that is to be obtained
parameter	UInt16	The parameter to obtain
value	double[ ]	The value that was read
return value	int	• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kUNKNOWNPARAMETERERROR: The parameter is not recognized • kNOERROR(0): The above errors did not occur

```
int error = aasp.GetFloatArrayParameter(device,
parameter, out value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Memory Commands

These commands are high-level commands used to perform reads and writes of the memory of a device. For these commands to succeed, the protocol must be set and the protocol parameters must match the expectations of the device.

A memory address is a 32-bit unsigned number, where the lower 24 bits are the address and the upper 8 bits are flags that describe the access. The flags are as follows:

Bit Name	Bit Number(s)	Description
WriteOnly	7	When true, the memory location is write only, it does not respond to a read
Reserved	6	Unused; reserved for future purposes
Reserved	5	Unused; reserved for future purposes
Reserved	4	Unused; reserved for future purposes
Reserved	3	Unused; reserved for future purposes
MemorySpace	2:0	0 = direct memory access 1 – 7 = indirect memory access group

Possible errors in all memory commands are:

- kINVALIDDEVICEERROR when the device is not listed in the device table of the programmer;
- kCOMMUNICATIONPROTOCOLNOTSETERROR when the communication protocol is not set; and
- kBADCOMMUNICATIONPROTOCOLERROR when the protocol does not support reads or writes of device memory.

Read(int, UInt32, out UInt32)

Reads the contents of a single address from the device.

Parameter	Type	Description
device	int	The device that contains the memory that is to be read
address	UInt32	The address to be read
value	UInt32	The value that was read
return value	int	• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.Read(device, address, out value);
```

Read(int, UInt32[ ], out UInt32[ ])

Reads the contents of the addresses from the device.

Parameter	Type	Description
device	int	The device that contains the memory that is to be read
addresses	UInt32[ ]	The addresses to be read
values	UInt32[ ]	The values that were read
return value	int	• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.Read(device, addresses, out values);
```

Read(int, UInt32[ ], out UInt32[ ], out int[ ])

Reads the contents of the addresses from the device, and returns an error for each address read.

Parameter	Type	Description
device	int	The device that contains the memory that is to be read
addresses	UInt32[ ]	The addresses to be read
values	UInt32[ ]	The values that were read
errors	int[ ]	The individual error codes that occurred during the address reads
return value	int	• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.Read(device, addresses, out values,
out errors);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Read(int, UInt32, UInt32, out UInt32[ ])

Reads the range of addresses from the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be read
start_address	UInt32	The first address to be read
end_address	UInt32	The last address to be read
values	UInt32[ ]	The values that were read
return value	int	• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.Read(device, start_address, end_
address, out values);
```

Reads a field from the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be read
address	UInt32	The address to be read
high_bit	int	The highest bit location in the field
low_bit	int	The lowest bit location in the field
value	UInt32	The value that was read
return value	int	• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.ReadField(device, address, high_bit,
low_bit, out value);
```

ReadField(int, UInt32[ ], int[ ], int[ ], out UInt32[ ])

Reads the fields from the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be read
addresses	UInt32[ ]	The addresses to be read
high_bits	int[ ]	The highest bit locations in the field
low_bits	int[ ]	The lowest bit locations in the field
values	UInt32[ ]	The values that were read
return value	int	• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.ReadField(device, addresses, high_
bits, low_bits, out values);
```

ReadField(int, UInt32[ ],int[ ], int[ ], out UInt32[ ], out int[ ])

Reads the addresses from the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be read
addresses	UInt32[ ]	The addresses to be read
high_bits	int[ ]	The highest bit locations in the field
low_bits	int[ ]	The lowest bit locations in the field
values	UInt32[ ]	The values that were read
errors	int[ ]	The individual error codes that occurred when reading the field
return value	int	• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.ReadField(int device, addresses,
high_bits, low_bits, out values, out errors);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Write(int, UInt32, UInt32)

Writes the value to the address on the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
address	UInt32	The address to be written
value	UInt32	The value that is to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.Write(device, address, value);
```

Write(int, UInt32[ ], UInt32[ ])

Writes the values to the addresses on the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
values	UInt32[ ]	The values that are to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.Write(device, addresses, values);
```

Write(int, UInt32[ ], UInt32[ ], out int[ ])

Writes the values to the addresses on the device; errror does not throw an exception.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
values	UInt32[ ]	The values that are to be written
errors	int[ ]	The individual error codes that occurred during the read of the addresses
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.Write(device, addresses, values, out
errors);
```

WriteField(int, UInt32, int, int, UInt32)

Writes the value to the field on the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
address	UInt32	The address to be written
high_bit	int	The highest bit location in the field
low_bit	int	The lowest bit location in the field
value	UInt32	The value that is to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteField(device, address, high_
bit, low_bit, value)
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

WriteField(int, UInt32[ ],int[ ], int[ ], UInt32[ ])

Writes the values to the field on the device.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
high_bits	int[ ]	The highest bit locations in the fields
low_bits	int[ ]	The lowest bit locations in the fields
values	UInt32[ ]	The value that is to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteField(device, addresses, high_
bits, low_bits, values)
```

WriteField(int, UInt32[ ],int[ ], int[ ], UInt32[ ], out int[ ])

Writes the values to the fields on the device; an error does not throw an exception.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
high_bits	int[ ]	The highest bit locations in the fields
low_bits	int[ ]	The lowest bit locations in the fields
values	UInt32[ ]	The value that is to be written
errors	int[ ]	The individual error codes that occurred during the write to the field
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteField(device, addresses, high_
bits, low_bits, values, out errors);
```

WriteVerify(int, UInt32, UInt32)

Parameter	Type	Description
device	int	The device that has the memory that is to be written
address	UInt32	The address to be written
value	UInt32	The value that is to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOTVERIFIEDERROR: The verification operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteVerify(device, address, value);
```

WriteVerify(int, UInt32[ ], UInt32[ ])

Writes the values to the addresses on the device, and verifies that values are correctly written.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
values	UInt32[ ]	The values that are to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOTVERIFIEDERROR: The verification operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteVerify(device, addresses,
values);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

WriteVerify(int, UInt32[ ], UInt32[ ], out int[ ])

Writes the values to the addresses on the device, and verifies the values are correctly written.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
values	UInt32[ ]	The values that are to be written
errors	int[ ]	The individual error codes that occurred during the write to or verification of the field
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOTVERIFIEDERROR:The verification operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteVerify(device, addresses,
values, out errors);
```

WriteFieldVerify(int, UInt32, int, int, UInt32)

Writes the values to the fields on the device, and verifies the values are correctly written.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
address	UInt32	The address to be written
high_bit	int	The highest bit location in the field
low_bit	int	The lowest bit location in the field
value	UInt32	The value that is to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOTVERIFIEDERROR:The verification operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteFieldVerify(device, address,
high_bit, low_bit, value)
```

WriteFieldVerify(int, UInt32[ ], byte[ ], byte[ ], UInt32[ ])

Writes the values to the fields on the device, and verifies the values are correctly written.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
high_bits	int[ ]	The highest bit locations in the field
low_bits	int[ ]	The lowest bit locations in the field
values	UInt32[ ]	The values that are to be written
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOTVERIFIEDERROR:The verification operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteFieldVerify(device, addresses,
high_bits, low_bits, values);
```

WriteFieldVerify(int, UInt32[ ], byte[ ], byte[ ], UInt32[ ], out int[ ])

Writes the values to the fields on the device, and verifies the values are correctly written.

Parameter	Type	Description
device	int	The device that has the memory that is to be written
addresses	UInt32[ ]	The addresses to be written
high_bits	int[ ]	The highest bit locations in the field
low_bits	int[ ]	The lowest bit locations in the field
values	UInt32[ ]	The values that are to be written
errors	int[ ]	The individual error codes that occurred during the write to or verification of the field
return value	int	• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR: The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed • kNOTVERIFIEDERROR:The verification operation failed • kNOERROR(0): The above errors did not occur

```
int error = aasp.WriteFieldVerify(device, addresses,
high_bits, low_bits, values, out errors)
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Pulse Commands

SendPulseSequence(int, bool, int[ ], out double)

Sends a pulse sequence, and reads the device output, if desired.

Parameter	Type	Description
device	int	The device to which the pulses are to be sent
read_result	bool	If true, the device is read after all the pulses are sent
indexes	int[ ]	An array of indices in the voltage array
results	double	The value of the device, if requested
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SendPulseSequence(device, read_
result, indexes, out results);
```

SendPulse(int, bool, double, double, out double)

Sends a pulse, and reads the device output, if desired.

Parameter	Type	Description
device	int	The device to which the pulses are to be sent
read_result	bool	If true, the device is read after all the pulses are sent
voltage	double	The voltage of the pulse, in V
width	double	The width of the pulse ,in seconds
results	double	The value of the device, if requested
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SendPulse(device, read_result,
voltage, width, out results);
```

SendProgramPulses(int)

Sends the program pulses.

Parameter	Type	Description
device	int	The device to which the pulses are to be sent
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SendProgramPulses(device);
```

SendPulsesReturnResponses(int, int, int, double, out bool[ ])

Sends the pulses, and returns a value of true or false.

Parameter	Type	Description
device	int	The device to which the pulses are to be sent
count	int	The number of pulses to send
pulse_value	int	The index into the voltage array
sample_ width	double	The width of the time between pulses, where sampling occurs
results	bool[ ]	The values of the sampling
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SendPulsesReturnResponses(device,
count, pulse_value, sample_width, out results);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Scan Vector Commands

ScanVector(int, int, byte[ ], out byte[ ])

Performs a scan vector.

Parameter	Type	Description
device	int	The device on which the vector is run
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
results	byte[ ]	The results of the scan vector; if an error occurs, this field might be null
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ScanVector(device, count, vector,
out results);
```

LoadScanVector(int, int, int, byte[ ])

Loads a scan vector.

Parameter	Type	Description
device	int	The device on which the vector is run
vector_id	int	The scan vector identifier
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.LoadScanVector(port, vector_id,
count, vector);
```

DeleteScanVector(int, int)

Deletes a scan vector.

Parameter	Type	Description
device	int	The device on which the vector is run
vector_id	int	The scan vector identifier
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.DeleteScanVector(device, vector_id)
```

TransitionVector(int, int, byte[ ])

Performs a transition vector.

Parameter	Type	Description
device	int	The device on which the vector is run
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.TransitionVector(device, count,
vector);
```

LoadTransitionVector(int, int, int, byte[ ])

Loads a transition vector.

Parameter	Type	Description
device	int	The device on which the vector is run
vector_id	int	The vector identifier
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.LoadTransitionVector(device, vector_
id, count, vector);
```

DeleteTransitionVector(int, int)

Deletes a transition vector.

Parameter	Type	Description
device	int	The device on which the vector is run
vector_id	int	The vector identifier
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.DeleteTransitionVector(device,
vector_id)
```

RunVectorSequence(int, int[ ])

Runs a scan-and-transition vector sequence.

Parameter	Type	Description
device	int	The device on which the vectors are run
sequence	int[ ]	The array of vector identifiers
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.RunVectorSequence(device, sequence);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Read, Output Commands

ReadDeviceOutputVoltage(int, out double)

Reads the output voltage from the device.

Parameter	Type	Description
device	int	The device that has an output that is to be read
voltage	double	The voltage from V_{OUT}, in V
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDeviceOutputVoltage(device, out voltage);
```

ReadDeviceOutputCurrent(int, out double)

Reads the output current from the device.

Parameter	Type	Description
device	int	The device that has an output that is to be read
current	double	The current from V_{CC}, in A
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDeviceOutputCurrent(device, out current);
```

ReadDeviceOutputDigital(int, out uint)

Reads the output digital value from the device.

Parameter	Type	Description
device	int	The device that has an output that is to be read
digital_ value	int	The digital value from the device
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDeviceOutputDigital(device, out digital_value);
```

ReadDeviceOutputPWM(int, out double, out double)

Reads the output PWM from the device.

Parameter	Type	Description
device	int	The device that has an output that is to be read
duty_cycle	double	The duty cycle of the PWM, as a percentage
frequency	double	The frequency of the PWM, in Hz
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDeviceOutputPWM(device, out duty_cycle, out frequency)
```

ReadDeviceOutputSENT(int, out uint)

Reads the output SENT message from the device, including all nibbles except the CRC nibble.

Parameter	Type	Description
device	int	The device that has an output that is to be read
message	uint	The SENT message from the device
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDeviceOutputSENT(device, out message)
```

ReadDeviceOutputSENTSlowSerialData(int, out uint)

Reads the output SENT slow serial data from the device.

Parameter	Type	Description
device	int	The device that has an output that is to be read
data	uint	The SENT slow serial data from the device
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadDeviceOutputSENTSlowSerialData(device, out data);
```

SendDeviceOutputSENTTrigger(int, double)

Sends a SENT trigger to the device.

Parameter	Type	Description
device	int	The device that has an output that is to be read
trigger_ width	double	The width of the SENT trigger, in seconds
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SendDeviceOutputSENTTrigger(device, trigger_width);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Sample, Output Commands

SampleDeviceOutputVoltage(int, int, double)

Samples the voltage output of the device until the specified number of samples is collected.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SampleDeviceOutputVoltage(device,
samples, rate);
```

SampleDeviceOutputCurrent(int, int, double)

Samples the current output of the device until the specified number of samples is collected.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SampleDeviceOutputCurrent(device,
samples, rate);
```

SampleDeviceOutputDigital(int, int, double)

Samples the digital output of the device until the specified number of samples is collected.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SampleDeviceOutputDigital(device,
samples, rate);
```

SampleDeviceOutputPWM(int, int, double)

Samples the duty cycle and frequency output of the device until the specified number of samples is collected.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SampleDeviceOutputPWM(device,
samples, rate);
```

SampleDeviceOutputSENT(int, int, double)

Samples the SENT output of the device until the specified number of samples is collected.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SampleDeviceOutputSENT(device,
samples, rate);
```

SampleDeviceOutputSENTSlowSerialData(int, int, double)

Samples the SENT slow serial data output of the device until the specified number of samples is collected.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SampleDeviceOutputSENTSlowSerialData
(device, samples, rate)
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SampleDeviceMemory(int, int, double, UInt32[ ])

Samples the memory locations of the device until the specified number of samples is collected.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
addresses	UInt32[ ]	The addresses to sample
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.SampleDeviceMemory(device, samples,
rate, addresses);
```

ReadSampleBufferVoltages(int, out double[ ], out bool)

Reads the sample buffer of the device for voltage.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	double[ ]	An array of voltage samples
more_samples	bool	If there are more samples in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadSampleBufferVoltages(device, out
samples, out more_samples);
```

ReadSampleBufferCurrents(int, out double[ ], out bool)

Reads the sample buffer of the device for current.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	double[ ]	An array of current samples
more_samples	bool	If there are more samples in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadSampleBufferCurrents(device, out
samples, out more_samples);
```

ReadSampleBufferDigital(int, out int[ ], out bool)

Reads the sample buffer of the device for digital output.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	int[ ]	An array of digital samples
more_samples	bool	If there are more samples in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadSampleBufferDigital(int device,
out uint[] value_samples, out bool more_samples)
```

ReadSampleBufferPWM(int, out double[ ], out double[ ], out bool)

Reads the sample buffer of the device for PWM.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
duty_cycles	double[ ]	An array of duty cycles from the samples
frequencies	double[ ]	An array of frequencies from the samples
more_samples	bool	If there are more samples in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadSampleBufferPWM(device, out
duty_cycles, out frequencies, out more_samples);
```

ReadSampleBufferSENT(int, out uint[ ], out bool)

Reads the sample buffer of the device for SENT messages.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	uint[ ]	An array of SENT messages
more_samples	bool	If there are more samples in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadSampleBufferSENT(device, out
samples, out more_samples);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ReadSampleBufferSENTSSD(int, out uint[ ], out bool)

Reads the sample buffer of the device for SENT slow serial data.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	uint[ ]	An array of SENT slow serial data
more_samples	bool	If there are more samples in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadSampleBufferSENTSSD(device, out
samples, out more_samples);
```

ReadSampleBufferMemory(int, out uint[ ], out bool)

Reads the sample buffer of the device.

Parameter	Type	Description
device	int	The device that has an output that is to be sampled
samples	uint[ ]	An array of memory location contents
more_samples	bool	If there are more samples in the buffer, the value is true
return value	int	If the call fails, returns an error; otherwise, returns kNOERROR(0)

```
int error = aasp.ReadSampleBufferMemory(device, out
samples, out more_samples);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

AASP_PORT COMMANDS

The commands that operate on a port are grouped into the AASP_Port class. This grouping prevents the need to retain and pass the port number as a parameter for each command. For any command in this class, if an error occurs, an error code is not returned; rather, an exception is thrown with the error included.

Port Power Commands

SetVccOn(double, bool)

Sets the VCC port to the desired voltage.

Parameter	Type	Description
voltage	double	The voltage at which V_{CC} is to be set
unlock_device	bool	If the access codes are to be sent to the devices, the value is true
		If the call fails, an exception is thrown

```
aasp_port.SetVccOn(voltage, unlock_device);
```

SetVccOff( )

Performs turn-off of the port.

Parameter	Type	Description
		If the call fails, an exception is thrown

```
aasp_port.SetVccOff();
```

GetVcc( )

Obtains the voltage on the V_{CC} line of the port.

Parameter	Type	Description
return value	double	The actual voltage of V_{CC}
		If the call fails, an exception is thrown

```
double voltage = aasp_port.GetVcc();
```

GetIcc( )

Obtains the current being supplied to the port.

Parameter	Type	Description
return value	double	The actual current being supplied on V_{CC}, in mA
		If the call fails, an exception is thrown

```
double current = aasp_port.GetIcc();
```

Port Feature Values and Commands

The different port features that can be set or obtained from the port. Not all features are available on all ports or programmers. If a feature is not available on the port or programmer a kUNKNOWNPARAMETERERROR exception is thrown. If the value given to set a feature is not within the range of valid values, a kBADPARAMETERVALUEERROR exception is thrown. When a feature is set, the hardware updates to reflect the new value.

Port Feature Values

Name	Feature ID	Description
PortVoutPull-upSelection	0	Pull-up used on V_{OUT} (uint): 0 = off > 0 = hardware specific
PortVoutPull-upVoltage	1	Voltage used on V_{OUT} pull-up (float)
PortInputHighThreshold	2	Input high-threshold voltage (float)
PortInputLowThreshold	3	Input low-threshold voltage (float)
PortGPIOVoltage	4	GPIO voltage (float)
PortGPIOPull-upSelection	5	Pull-up used on the GPIO bus (uint): 0 = off > 0 = hardware specific
PortI2CVoltage	6	I^2C bus voltage (float)
PortI2CPull-upSelection	7	Pull-up used on the I^2C bus (uint): 0 = off > 0 = hardware specific
PortSPIVoltage	8	SPI bus voltage (float)
PortVoutPull-downSelection	9	Pull-down used on V_{OUT} (uint): 0 = off > 0 = hardware specific
PortVoutCapacitorSelection	10	Capacitor used on V_{OUT}: 0 = off > 0 = hardware specific (uint)
PortVccBypassCapacitorEnable	11	Bypass capacitor used on V_{CC} (bool)
PortVccRisingSlewRate	12	Slew rate for the rising V_{CC}, in V/ms (float)
PortVccFallingSlewRate	13	Slew rate for the falling V_{CC}, in V/ms (float)

ALLEGRO microsystems
Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SetPortFeature(UInt16, UInt32)

Sets the feature of a port.

Parameter	Type	Description
feature	UInt16	The port feature to change
value	UInt32	The value to write
		If the call fails, an exception is thrown

```
aasp_port.SetPortFeature(feature, value);
```

GetPortFeature(UInt16)

Obtains the feature from a port.

Parameter	Type	Description
feature	UInt16	The port feature to change
return value	UInt32	The value that was read
		If the call fails, an exception is thrown

```
UInt32 value = aasp_port.GetPortFeature(feature);
```

SetPortFloatFeature(UInt16, double)

Sets the float feature of a port.

Parameter	Type	Description
feature	UInt16	The port feature to change
value	double	The value to write
		If the call fails, an exception is thrown

```
aasp_port.SetPortFloatFeature(feature, value);
```

GetPortFloatFeature(UInt16)

Obtains the feature from a port.

Parameter	Type	Description
feature	UInt16	The port feature to change
return value	double	The value that was read
		If the call fails, an exception is thrown

```
double value = aasp_port.
GetPortFloatFeature(feature);
```

Port Properties

All of the port features can be accessed through properties of the AASP_Port class. This means that they can be read and written as if they were variables.

PortVoutPull-upSelection

Selects the pull-up used on V_{OUT}.

```
aasp_port.PortVoutPull-upSelection = 1;
```

PortVoutPull-upVoltage

Voltage used on the V_{OUT} pull-up.

```
aasp_port.PortVoutPull-upVoltage = 5.0;
```

PortInputHighThreshold

High-threshold voltage of the input.

```
aasp_port.PortInputHighThreshold = 2.5;
```

PortInputLowThreshold

Low-threshold voltage of the input.

```
aasp_port.PortInputLowThreshold = 1.0;
```

PortGPIOVoltage

Voltage used on the GPIO drivers and pull-ups.

```
aasp_port.PortGPIOVoltage = 5.0;
```

PortGPIOPull-upSelection

Selection of pull-up used on the GPIOs.

```
aasp_port.PortGPIOPull-upSelection
```

PortI2CVoltage

Voltage used on the I^2C drivers and pull-ups.

```
aasp_port.PortI2CVoltage = 3.3;
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

PortI2CPull-upSelection

Selection of pull-up used on the I^2C bus.

```
aasp_port.PortI2CPull-upSelection = 1;
```

PortSPIVoltage

Voltage used on the SPI drivers.

```
aasp_port.PortSPIVoltage = 5.0;
```

PortVoutPull-downSelection

Selection of pull-down used on V_{OUT}.

```
aasp_port.PortVoutPull-downSelection = 1;
```

PortVoutCapacitorSelection

Selection of capacitor used on V_{OUT}.

```
aasp_port.PortVoutCapacitorSelection = 0;
```

PortVccBypassCapacitorEnable

Selection of bypass capacitor used on V_{CC}.

```
aasp_port.PortVccBypassCapacitorEnable = false;
```

PortVccRisingSlewRate

Slew rate of the rising V_{CC}, in V/ms.

```
aasp_port.PortVccRisingSlewRate = 1.0;
```

PortVccFallingSlewRate

Slew rate of the falling V_{CC}, in V/ms.

```
aasp_port.PortVccFallingSlewRate = 1.0;
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

GPIO Port Commands

SetPortBitsDirections(UInt32)

Sets the direction of the port bits (1 = output; 0 = input).

Parameter	Type	Description
value	UInt32	The direction of the port GPIO bits (1 = output; 0 = input)
		If the call fails, an exception is thrown

```
aasp_port.SetPortBitsDirections(value);
```

WritePortOutputBits(UInt32)

Writes the values to the output bits.

Parameter	Type	Description
value	UInt32	The value to be written to the GPIO bits
		If the call fails, an exception is thrown

```
aasp_port.WritePortOutputBits(value);
```

SetPortOutputBits(UInt32)

Sets the bits in the output to 1 that are set to 1 in the mask; does not set any bits that are set to 0 in the mask.

Parameter	Type	Description
mask	UInt32	The mask of the bits in the GPIO that are to be set: 1 = set; 0 = no action
		If the call fails, an exception is thrown

```
aasp_port.SetPortOutputBits(mask);
```

ClearPortOutputBits(UInt32)

Sets the bits in the output to 0 that are set to 1 in the mask; does not clear any bits that are set to 0 in the mask.

Parameter	Type	Description
mask	UInt32	The mask of the bits in the GPIO that are to be cleared: 1 = cleared; 0 = no action
		If the call fails, an exception is thrown

```
aasp_port.ClearPortOutputBits(mask);
```

TogglePortOutputBits(UInt32)

Toggles the bits in the output that are set to 1 in the mask; does not toggle any bits that are set to 0 in the mask.

Parameter	Type	Description
mask	UInt32	The mask of the bits in the GPIO that are to be toggled: 1 = toggled; 0 = no action
		If the call fails, an exception is thrown

```
aasp_port.TogglePortOutputBits(mask);
```

ReadPortOutputBits( )

Reads the state of the output bits, and returns the values.

Parameter	Type	Description
return value	UInt32	The state of the output bits
		If the call fails, an exception is thrown

```
UInt32 value = aasp_port.ReadPortOutputBits();
```

ReadPortInputBits( )

Reads the input bits, and returns the values.

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

AASP_DEVICE COMMANDS

The commands that operate on a device are grouped into the AASP_Device class. This grouping prevents the need to remember and pass the device ID as a parameter to the methods. Each method in this class does not return an error code; rather, an exception is thrown with the error in it.

Device Commands

GetProgrammer( )

Returns the programmer to which the device is attached.

Parameter	Type	Description
return value	AASP	The AASP object to which the device is connected
		If the call fails, an exception is thrown

```
AASP aasp = aasp_device.GetProgrammer ();
```

GetPort( )

Returns the port to which the device is attached.

Parameter	Type	Description
return value	AASP_ PORT	The AASP_Port object to which the device is connected
		If the call fails, an exception is thrown

```
AASP_Port aasp_port = aasp_device.GetPort ();
```

GetDeviceID( )

Returns the identifier of the device.

Parameter	Type	Description
return value	int	The identifier of the device
		If the call fails, an exception is thrown

```
int device_id = aasp_device.GetDeviceID ();
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Parameters

The device is a construct that contains the communication parameters needed to talk to a physical device. The parameter commands are used to change those parameters. Because a device might need to communicate via different protocols at the same time, storage is allocated for all protocols. Not all programmers implement all parameters.

Device Parameter Values

Name	Value	Description
DeviceProtocol	0	Protocol used by the device (uint): 0 = No protocol 1 = SPI 2 = I^2C 3 = Manchester 4 = Pulses
DeviceDelayAfterEepromWrite	1	Delay between EEPROM writes, in seconds (float)
DeviceOutputType	2	Output of the device (uint): 0 = No output 1 = Analog 2 = PWM 3 = SENT 4 = Other
DeviceDelayAfterPowerOn	3	Delay before commands at power-up, in seconds (float)
DeviceNeedPowerOnUnlock	4	Unlock required at power-up (bool)
DeviceUnlockAddresses	5	Addresses of the unlock register (uint)
DeviceUnlockCodes	6	Unlock code (uint[ ])
DeviceUnlockCodesWidth	7	Width of unlock code (uint)
DeviceNumberOfIndirectMemories	8	Device number of indirect memories (uint)
DeviceHasCommunicationEnable	9	Device has communication enable (bool)
DeviceCommunicationEnableAddress	10	Device communication-enable address (uint)
DeviceCommunicationEnableMask	11	Device communication-enable mask (uint)
DeviceCommunicationEnableActiveState	12	Device communication-enable active state (bool)
DeviceCommunicationEnableInAccessCode	13	Device communication enable is the lsb of the access code (bool)
DeviceProgramPulsesRequired	14	Device requires programming pulses (uint)
DeviceProgramPulseVoltage	15	Voltage level for program pulses, in V (float)
DeviceProgramPulseWidth	16	Width of program pulses, in seconds (float)
DeviceProgramPulseDelay	17	Width of delay between program pulses, in seconds (float)
DeviceProgramPulseOutputType	18	Location where the program pulse is applied (uint): 0 = No program pulses 1 = Program pulses on V_{CC} 2 = Program pulses on V_{OUT} 3 = Program pulses on other
DevicePrologueWriteAddresses	19	Location where write occurs during the prologue (uint[ ])
DevicePrologueWriteValues	20	Data to be written during the prologue (uint[ ])
DevicePrologueReadAddresses	21	Location where read occurs during the prologue (uint[ ])
DevicePrologueReadValues	22	Data to read during the prologue (uint[ ])
DevicePrologueReadMasks	23	Data to mask from the read during the prologue (uint[ ])
DeviceEpilogueWriteAddresses	24	Location where a write occurs during the epilogue (uint[ ])
DeviceEpilogueWriteValues	25	Data to be written during the epilogue (uint[ ])

Continued on next page...

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Parameter Values ... *continued from previous page*

Name	Value	Description
DeviceLowestEEPROMAddress	26	The lowest EEPROM address (uint)
DeviceHighestEEPROMAddress	27	The highest EEPROM address (uint)
DeviceName	28	Device name (string)
DeviceInformation	29	Device information (string)
DeviceDisableEEPROMWrites	30	Disable writing to the device EEPROM (uint)
DeviceOutputReadAddress	31	Location where read occurs for the digital output (uint)
DeviceOutputIsSigned	32	Indicates if the digital output is signed (bool)
DeviceOutputReadMask	33	Indicates what bits to read for the digital output (uint)
DeviceIndirectReadAddressRegister	34	The indirect read address (uint)
DeviceIndirectReadAddressMask	35	The read address mask (uint)
DeviceIndirectReadDataRegister	36	Read-data register (uint)
DeviceIndirectReadDataWidth	37	Width of the data to be read (uint)
DeviceIndirectReadDataDirection	38	If the msb is the lowest address, the value is true (bool)
DeviceIndirectReadControlRegister	39	Read-data control register (uint)
DeviceIndirectReadCommand	40	Read-data command (uint[ ])
DeviceIndirectReadStatusRegister	41	Read-data status register (uint)
DeviceIndirectReadStatusMask	42	Read-data status mask (uint[ ])
DeviceIndirectReadStatus	43	Read-data status (uint[ ])
DeviceIndirectWriteAddressRegister	44	The indirect-write address (uint)
DeviceIndirectWriteAddressMask	45	The write-address mask (uint)
DeviceIndirectWriteDataRegister	46	The write-data register (uint)
DeviceIndirectWriteDataWidth	47	Width of the data to be written (uint)
DeviceIndirectWriteDataDirection	48	If the msb is the lowest address, the value is true (bool)
DeviceIndirectWriteControlRegister	49	The write-data control register (uint)
DeviceIndirectWriteCommand	50	The write-data command (uint[ ])
DeviceIndirectWriteStatusRegister	51	The write-data status register (uint)
DeviceIndirectWriteStatusMask	52	The write-data status mask (uint[ ])
DeviceIndirectWriteStatus	53	The write-data status (uint[ ])
DeviceIndirectMemoryWidth	54	Width of the indirect memory to be read or written (uint)
DeviceDirectMemoryWidth	55	Width of a data register to be directly read or written (uint)
DeviceProgramPulseRisingSlewRate	56	Slew rate of the rising program pulse, in V/ms (float)
DeviceProgramPulseFallingSlewRate	57	Slew rate of the falling program pulse, in V/ms (float)

I²C Parameter Values

Name	Value	Description
I2CClockFrequency	1000	I²C clock frequency, in Hz (uint)
I2CAddress	1001	I²C address of the device (uint)
I2CRegisterWidth	1002	Size of the register read or write (uint)
I2CByteAddressing	1003	Indicates if the device has byte addressing (uint)
I2CHasCRC	1004	Indicates if the device has a CRC (bool)

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Manchester Parameter Values

Name	Value	Description
ManchesterSyncSize	2000	Size of the initial synchronization, in bits (uint)
ManchesterCommandSize	2001	Size of the command, in bits (uint)
ManchesterIDSize	2002	Size of the ID, in bits (uint)
ManchesterFieldSelectSize	2003	Size of the field select, in bits (uint)
ManchesterAddressSize	2004	Size of the address, in bits (uint)
ManchesterReadDataSize	2005	Size of the read data, in bits (uint)
ManchesterWriteDataSize	2006	Size of the write data, in bits (uint)
ManchesterStatusSize	2007	Size of the status in a read response, in bits (uint)
ManchesterCRCSize	2008	Size of the CRC, in bits (uint)
ManchesterCommandOutput	2009	Locations where Manchester pulses are applied and responses are received (uint): 0 = No Manchester 1 = V_{CC}, responses on V_{OUT} 2 = Analog V_{OUT}, responses on V_{OUT} 3 = Digital open-drain V_{OUT}, responses on V_{OUT}; requires a pull-up 4 = Digital push-pull V_{OUT}, responses on V_{OUT} 5 = Current, command on V_{CC}, responses on I_{CC} 6 = Other
ManchesterCommandEnableType	2010	Which method is needed for enabling communication with the device (uint): 0 = None 1 = V_{CC} overvoltage 2 = Overdrive 3 = PWM function pulse 4 = SENT function pulse 5 = Trigger SENT and function pulse 6 = Auxiliary function pulse 7 = Assert GPIO 12 = V_{CC} overvoltage and overdrive 13 = V_{CC} overvoltage and PWM function pulse 14 = V_{CC} overvoltage and SENT function pulse 15 = V_{CC} overvoltage and trigger SENT and function pulse 16 = V_{CC} overvoltage and auxiliary function pulse
ManchesterDelayAfterCommandEnable	2011	Delay before first command after raising V_{CC} to command-enable voltage, in seconds (float)
ManchesterCommandEnableVccVoltage	2012	Voltage to which V_{CC} is to be raised when performing a command-enable operation (float)
ManchesterCommandEnableVOutLevel	2013	Logical state to which V_{OUT} is to be raised when performing a command-enable operation (bool)
ManchesterHandlesMultipleCommands	2014	If multiple commands can be sent without the need for a program-enable cycle, set value to true (bool)
ManchesterHasWriteResponse	2015	If the device has a write response, the value is true; only useful for EEPROM writes (bool)
ManchesterWriteResponseLevel	2016	Logical state to which the output transitions for the write response: 0 = high to low; 1 = low to high (bool)
ManchesterBitRate	2017	Bit rate, in bits per second (uint)
ManchesterSlewRate	2018	Slew rate, in V/ms (float)
ManchesterHighLevelVoltage	2019	Manchester high voltage (float)
ManchesterLowLevelVoltage	2020	Manchester low voltage (float)
ManchesterTriggerWidth	2021	Width of trigger, in seconds (float)
ManchesterDelayWidth	2022	Delay after trigger, in seconds (float)
ManchesterAuxPulseWidth	2023	Width of auxiliary pulse, in seconds (float)
ManchesterHighDelayWidth	2024	Width of high-voltage delay, in seconds (float)

Continued on next page...

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Manchester Parameter Values ... *continued from previous page*

Name	Value	Description
ManchesterLowDelayWidth	2025	Width of low-voltage delay, in seconds (float)
ManchesterBitTimesBeforeCommand	2026	Number of bit times before the command (uint)
ManchesterBitTimesAfterCommand	2027	Number of bit times after the command (uint)
ManchesterDelayAfterRead	2028	Delay after read, before next command, in seconds (float)
ManchesterDelayAfterWrite	2029	Delay after write, before next command, in seconds (float)
ManchesterDelayAfterEEPROMWrite	2030	If device does not have write response, delay after EEPROM write, in seconds (float)
ManchesterCommandRetries	2031	If a read fails or a write does not receive a write response, number of command retries (uint)
ManchesterNeedPull-upForRead	2032	If the device needs a pull-up during a read operation, set to true; used when the device typically has a push-pull driver; except when responding to commands, it has an open-drain driver (bool)
ManchesterID	2100	Manchester ID (uint)
ManchesterFieldSelect	2101	Field select value (uint)

SPI Parameter Values

Name	Value	Description
SPIClockFrequency	3000	SPI clock frequency, in Hz (uint)
SPIMSBFirst	3001	SPI: If value is true, msb is first; otherwise, lsb is first (bool)
SPIMode	3002	SPI mode (0 through 3) (uint)
SPISaferSPI	3003	If the device is using SafeSPI, the value is true (bool)
SPITransferSize	3004	Width of each transfer, in number of bits (uint)
SPIWriteCommand	3005	Bit pattern of the write command (uint)
SPIReadCommand	3006	Bit pattern of the read command (uint)
SPICommandFieldSize	3007	Width of the command (uint)
SPICommandFieldShift	3008	The command shift (uint)
SPIAddressFieldSize	3009	Width of the address (uint)
SPIAddressFieldShift	3010	The address shift (uint)
SPIDataFieldSize	3011	Width of the data (uint)
SPIDataFieldShift	3012	The data shift (uint)
SPICRCFieldSize	3013	Width of the CRC (uint)

SENT Parameter Values

Name	Value	Description
SENTNumberOfNibbles	5000	Number of nibbles in the SENT message (uint)
SENTTickTime	5001	Tick time, in seconds (float)
SENTIncludeSCNInCRC	5002	Include the status and control nibble in the CRC (bool)
SENTType	5003	Type of SENT (uint): 0 = Streaming SENT 1 = Triggered SENT 2 = Addressable SENT 3 = Sequential SENT
SENTSensorID	5004	Sensor identifier (uint)
SENTMaxSensorID	5005	Maximum sensor identifier (uint)
SENTTriggerWidth	5006	Width of TSENT trigger, in ticks (uint)
SENTAuxPulseWidth	5007	Width of ASENT and SSENT F_AUX pulse, in ticks (uint)
SENTAddressPulseWidth	5008	Width of ASENT IncAdr pulse, in ticks; does not include the five-tick low period (uint)
SENTSlowSerialDataMode	5009	Slow serial data mode (uint): 0 = Standard 1 = Extended
SENTSyncPulseWidth	5010	Width of SSENT F_SYNC pulse, in ticks (uint)
SENTOutputPulseWidth	5011	Width of ASENT and SSENT F_OUTPUT pulse, in ticks (uint)
SENTSamplePulseWidth	5012	Width of ASENT and SSENT F_SAMPLE pulse, in ticks (uint)

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Pulse Parameter Values

Name	Value	Description
PulseOutput	6000	Locations where pulses are applied and responses are received (uint): • 0 = No pulses, no responses • 1 = V_{CC}, response on V_{OUT} • 2 = V_{CC}, response on Icc • 3 = V_{OUT}, response on V_{OUT} • 4 = Other
PulseOnWidth	6001	Set width of pulse-on operation, in seconds (float)
PulseOffWidth	6002	Set width of pulse-off operation, in seconds (float)
PulseVoltageLevels	6003	Voltage levels (float[ ])
PulseRisingSlewRate	6004	Slew rate of the rising pulses, in V/ms (float)
PulseFallingSlewRate	6005	Slew rate of the falling pulses, in V/ms (float)

Scan Vector Parameter Values

Name	Value	Description
SCClockFrequency	9000	Scan-vector clock frequency, in Hz (uint)
SVClockPolarity	9001	Clock active polarity for scan vector (bool): • false = active high • true = active low
SVClockPhase	9002	Clock phase for scan vector (bool): • false = data are captured upon the rising edge and output upon falling edge • true = data are captured upon the falling edge and output upon the rising edge
SVOutputPolarity	9003	Output polarity (bool): • false = active low • true = active high
SVEnablePolarity	9004	Scan-enable polarity (bool): • false = active low • true = active high
SVResetBit	9005	GPIO bit used for reset (uint)

Device Parameter Commands

SetParameter(UInt16, UInt32)

Sets the parameter of a device.

Parameter	Type	Description
parameter	UInt16	The parameter to change
value	UInt32	The value to write
		An exception is thrown for any of the following conditions:: • The device is not listed in the device table of the programmer • The parameter is not recognized • The parameter value is not within the range of permitted values

```
aasp_device.SetParameter(parameter, value);
```

GetParameter(UInt16)

Obtains the parameter from a device.

Parameter	Type	Description
parameter	UInt16	The parameter to obtain
return value	UInt32	The value that was read
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized

```
UInt32 value = aasp_device.GetParameter(parameter);
```

SetFloatParameter(UInt16, double)

Sets the float parameter of a device.

Parameter	Type	Description
parameter	UInt16	The parameter to change
value	double	The value to write
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized • The parameter value is not within the range of permitted values

```
aasp_device.SetFloatParameter(parameter, value);
```

GetFloatParameter(UInt16)

Obtains the parameter from a device

Parameter	Type	Description
parameter	UInt16	The parameter to obtain
return value	double	The value that was read
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized

```
double value = aasp_device.
GetFloatParameter(parameter);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SetStringParameter(UInt16, string)

Sets the parameter of a device.

Parameter	Type	Description
parameter	UInt16	The parameter to change
value	string	The value to write
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized • The parameter value is not within the range of permitted values

```
aasp_device.SetStringParameter(parameter, value);
```

GetStringParameter(UInt16)

Obtains the parameter from a device.

Parameter	Type	Description
parameter	UInt16	The parameter to obtain
return value	string	The value that was read
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized

```
string value = aasp_device.
GetStringParameter(parameter);
```

SetArrayParameter(UInt16, UInt32[ ])

Sets the array parameter of a device.

Parameter	Type	Description
parameter	UInt16	The parameter to change
value	UInt32[ ]	The value to write
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized • The parameter value is not within the range of permitted values

```
aasp_device.SetArrayParameter(parameter, values);
```

GetArrayParameter(UInt16)

Obtains the array parameter from a device.

Parameter	Type	Description
parameter	UInt16	The parameter to obtain
return value	UInt32[ ]	The value that was read
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized

```
UInt32[] value = aasp_device.
GetArrayParameter(parameter);
```

SetFloatArrayParameter(UInt16, double[ ])

Sets the float array parameter of a device.

Parameter	Type	Description
parameter	UInt16	The parameter to change
value	double[ ]	The value to write
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized • The parameter value is not within the range of permitted values

```
aasp_device.SetFloatArrayParameter(parameter,
values);
```

GetFloatArrayParameter(UInt16)

Obtains the float array parameter from a device.

Parameter	Type	Description
parameter	UInt16	The parameter to obtain
return value	double[ ]	The value that was read
		An exception is thrown for any of the following conditions: • The device is not listed in the device table of the programmer • The parameter is not recognized

```
double[] value = aasp_device.GetFloatArrayParameter(
parameter);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Properties

All device parameters are accessible through properties of the AASP_Device class. This means that they can be read and written as if they were variables.

DeviceProtocol

Protocol used by the device.

```
aasp_device.DeviceProtocol
```

DeviceDelayAfterEepromWrite

Delay between EEPROM writes in seconds.

```
aasp_device.DeviceDelayAfterEepromWrite
```

DeviceOutputType

Output of the device.

```
aasp_device.DeviceOutputType
```

DeviceDelayAfterPowerOn

Delay before commands at power-up, in seconds.

```
aasp_device.DeviceDelayAfterPowerOn
```

DeviceNeedPowerOnUnlock

Device needs to be unlocked during power-on sequence.

```
aasp_device.DeviceNeedPowerOnUnlock = true;
```

DeviceUnlockAddresses

Address of the unlock register (uint[]).

```
aasp_device.DeviceUnlockAddresses
```

DeviceUnlockCodes

Unlock codes (uint[]).

```
aasp_device.DeviceUnlockCodes
```

DeviceUnlockCodesWidth

Unlock codes width (uint).

```
aasp_device.DeviceUnlockCodesWidth
```

DeviceNumberOfIndirectMemories

Number of indirect memories of the device (uint).

```
aasp_device.DeviceNumberOfIndirectMemories
```

DeviceHasCommunicationEnable

Device has communication enable (bool).

```
aasp_device.DeviceHasCommunicationEnable
```

DeviceCommunicationEnableAddress

Device communication-enable address (uint).

```
aasp_device.DeviceCommunicationEnableAddress
```

DeviceCommunicationEnableMask

Device communication-enable mask (uint).

```
aasp_device.DeviceCommunicationEnableMask
```

DeviceCommunicationEnableActiveState

Device communication-enable active state (bool).

```
aasp_device.DeviceCommunicationEnableActiveState
```

DeviceCommunicationEnableInAccessCode

Device communication enable is the last bit of the access code (bool).

```
aasp_device.DeviceCommunicationEnableInAccessCode
```

DeviceProgramPulsesRequired

Device requires programming pulses (uint).

```
aasp_device.DeviceProgramPulsesRequired
```

DeviceProgramPulseVoltage

Voltage level for program pulses, in V (float).

```
aasp_device.DeviceProgramPulseVoltage
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

DeviceProgramPulseWidth

Width of the program pulses, in seconds (float).

```
aasp_device.DeviceProgramPulseWidth
```

DeviceProgramPulseDelay

Width of the delay between program pulses, in seconds (float).

```
aasp_device.DeviceProgramPulseDelay
```

DeviceProgramPulseOutputType

Location of application of the program pulse (uint)

```
aasp_device.DeviceProgramPulseOutputType
```

DevicePrologueWriteAddresses

Location where write occurs during the prologue (uint[]).

```
aasp_device.DevicePrologueWriteAddresses
```

DevicePrologueWriteValues

Values that are to be written during the prologue (uint[]).

```
aasp_device.DevicePrologueWriteValues
```

DevicePrologueReadAddresses

Location where read occurs during the prologue (uint[]).

```
aasp_device.DevicePrologueReadAddresses
```

DevicePrologueReadValues

Values that are to be read during the prologue (uint[]).

```
aasp_device.DevicePrologueReadValues
```

DevicePrologueReadMasks

Masks that are to be read during the prologue (uint[]).

```
aasp_device.DevicePrologueReadMasks
```

DeviceEpilogueWriteAddresses

Location where write occurs during the epilogue (uint[]).

```
aasp_device.DeviceEpilogueWriteAddresses
```

DeviceEpilogueWriteValues

Values that are to be written during the epilogue (uint[]).

```
aasp_device.DeviceEpilogueWriteValues
```

DeviceLowestEEPROMAddress

The lowest EEPROM address (uint).

```
aasp_device.DeviceLowestEEPROMAddress
```

DeviceHighestEEPROMAddress

The highest EEPROM address (uint).

```
aasp_device.DeviceHighestEEPROMAddress
```

DeviceName

The device name (string).

```
aasp_device.DeviceName
```

DeviceInformation

The device information (string).

```
aasp_device.DeviceInformation
```

DeviceDisableEEPROMWrites

Disable writing to the device EEPROM (uint).

```
aasp_device.DeviceDisableEEPROMWrites
```

DeviceOutputReadAddress

Location of digital output (uint).

```
aasp_device.DeviceOutputReadAddress
```

DeviceOutputIsSigned

If the digital output value is signed, the value is true (bool).

```
aasp_device.DeviceOutputIsSigned
```

DeviceOutputReadMask

Mask for the digital output (uint).

```
aasp_device.DeviceOutputReadMask
```

DeviceIndirectReadAddressRegister

The indirect read address (uint).

```
aasp_device.DeviceIndirectReadAddressRegister
```

DeviceIndirectReadAddressMask

The read-address mask (uint).

```
aasp_device.DeviceIndirectReadAddressMask
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

DeviceIndirectReadDataRegister

The read-data register (uint).

```
aasp_device.DeviceIndirectReadDataRegister
```

DeviceIndirectReadDataWidth

Width of the read-data register (uint).

```
aasp_device.DeviceIndirectReadDataWidth
```

DeviceIndirectReadDataDirection

If the msb is the lowest address, the value is true (bool).

```
aasp_device.DeviceIndirectReadDataDirection
```

DeviceIndirectReadControlRegister

The read-data control register (uint).

```
aasp_device.DeviceIndirectReadControlRegister
```

DeviceIndirectReadCommand

The read-data command (uint[]).

```
aasp_device.DeviceIndirectReadCommand
```

DeviceIndirectReadStatusRegister

The read-data status register (uint).

```
aasp_device.DeviceIndirectReadStatusRegister
```

DeviceIndirectReadStatusMask

The read-data status mask (uint[]).

```
aasp_device.DeviceIndirectReadStatusMask
```

DeviceIndirectReadStatus

The read-data status (uint[]).

```
aasp_device.DeviceIndirectReadStatus
```

DeviceIndirectWriteAddressRegister

The indirect-write address (uint).

```
aasp_device.DeviceIndirectWriteAddressRegister
```

DeviceIndirectWriteAddressMask

The write-address mask (uint).

```
aasp_device.DeviceIndirectWriteAddressMask
```

DeviceIndirectWriteDataRegister

The write-data register (uint).

```
aasp_device.DeviceIndirectWriteDataRegister
```

DeviceIndirectWriteDataWidth

Width of the data to be written (uint).

```
aasp_device.DeviceIndirectWriteDataWidth
```

DeviceIndirectWriteDataDirection

If the msb is the lowest address, the value is true (bool).

```
aasp_device.DeviceIndirectWriteDataDirection
```

DeviceIndirectWriteControlRegister

The write-data control register (uint).

```
aasp_device.DeviceIndirectWriteControlRegister
```

DeviceIndirectWriteCommand

The write-data command (uint[]).

```
aasp_device.DeviceIndirectWriteCommand
```

DeviceIndirectWriteStatusRegister

The write-data status register (uint).

```
aasp_device.DeviceIndirectWriteStatusRegister
```

DeviceIndirectWriteStatusMask

The write-data status mask (uint[]).

```
aasp_device.DeviceIndirectWriteStatusMask
```

DeviceIndirectWriteStatus

The write-data status (uint[]).

```
aasp_device.DeviceIndirectWriteStatus
```

DeviceIndirectMemoryWidth

The width of the indirect memory (uint).

```
aasp_device.DeviceIndirectMemoryWidth
```

DeviceDirectMemoryWidth

The width of the direct memory (uint).

```
aasp_device.DeviceDirectMemoryWidth
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

DeviceProgramPulseRisingSlewRate

The slew rate when the program pulse is rising (float).

```
aasp_device.DeviceProgramPulseRisingSlewRate
```

DeviceProgramPulseFallingSlewRate

The slew rate when the program pulse is falling (float).

```
aasp_device.DeviceProgramPulseFallingSlewRate
```

I2CClockFrequency

I²C clock freqency (uint).

```
aasp_device.I2CClockFrequency
```

I2CAddress

I²C device address on the I²C bus (uint).

```
aasp_device.I2CAddress
```

I2CRegisterWidth

Size of the register read or write (uint).

```
aasp_device.I2CRegisterWidth
```

I2CByteAddressing

Indicates if device is equipped for device byte addressing (bool).

```
aasp_device.I2CByteAddressing
```

I2CHasCRC

Indicates if device is equipped to return a CRC with the data (uint).

```
aasp_device.I2CHasCRC
```

ManchesterSyncSize

Size of the initial synchronization, in bits (uint).

```
aasp_device.ManchesterSyncSize
```

ManchesterCommandSize

Size of the command, in bits (uint).

```
aasp_device.ManchesterCommandSize
```

ManchesterIDSize

Size of the ID, in bits (uint).

```
aasp_device.ManchesterIDSize
```

ManchesterFieldSelectSize

Size of the field, in bits (uint).

```
aasp_device.ManchesterFieldSelectSize
```

ManchesterAddressSize

Size of the address, in bits (uint).

```
aasp_device.ManchesterAddressSize
```

ManchesterReadDataSize

Size of the read data, in bits (uint).

```
aasp_device.ManchesterReadDataSize
```

ManchesterWriteDataSize

Size of the write data, in bits (uint).

```
aasp_device.ManchesterWriteDataSize
```

ManchesterStatusSize

Size of the status of a read response, in bits (uint).

```
aasp_device.ManchesterStatusSize
```

ManchesterCRCSize

Size of the CRC, in bits (uint).

```
aasp_device.ManchesterCRCSize
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ManchesterCommandOutput

Location where the Manchester pulses are applied (uint).

```
aasp_device.ManchesterCommandOutput
```

ManchesterCommandEnableType

Method needed to enable communication with the device (uint).

```
aasp_device.ManchesterCommandEnableType
```

ManchesterDelayAfterCommandEnable

Delay before first command after raising V_{CC} to command-enable voltage (float).

```
aasp_device.ManchesterDelayAfterCommandEnable
```

ManchesterCommandEnableVccVoltage

Voltage to which V_{CC} is to be raised for performance of a command-enable operation (float).

```
aasp_device.ManchesterCommandEnableVccVoltage
```

ManchesterCommandEnableVOutLevel

Logical state to which V_{OUT} is to be raised for performance of a command-enable operation (uint).

```
aasp_device.ManchesterCommandEnableVOutLevel
```

ManchesterHandlesMultipleCommands

If multiple commands can be sent without the need to perform a program-enable cycle, the value is true (uint).

```
aasp_device.ManchesterHandlesMultipleCommands
```

ManchesterHasWriteResponse

If the device has a write response, the value is true (not 0) (uint).

```
aasp_device.ManchesterHasWriteResponse
```

ManchesterWriteResponseLevel

Logical state to which the output transitions for the write response (uint).

```
aasp_device.ManchesterWriteResponseLevel
```

ManchesterBitRate

Manchester bit rate (uint).

```
aasp_device.ManchesterBitRate
```

ManchesterSlewRate

Manchester slew rate (float).

```
aasp_device.ManchesterSlewRate
```

ManchesterHighLevelVoltage

Manchester high voltage (float).

```
aasp_device.ManchesterHighLevelVoltage
```

ManchesterLowLevelVoltage

Manchester low voltage (float).

```
aasp_device.ManchesterLowLevelVoltage
```

ManchesterTriggerWidth

Width of Manchester trigger (float).

```
aasp_device.ManchesterTriggerWidth
```

ManchesterDelayWidth

Width of Manchester delay (float).

```
aasp_device.ManchesterDelayWidth
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ManchesterAuxPulseWidth

Width of Manchester auxiliary pulse (float).

```
aasp_device.ManchesterAuxPulseWidth
```

ManchesterHighDelayWidth

Width of Manchester high-voltage delay (float).

```
aasp_device.ManchesterHighDelayWidth
```

ManchesterLowDelayWidth

Width of Manchester low-voltage delay (float).

```
aasp_device.ManchesterLowDelayWidth
```

ManchesterBitTimesBeforeCommand

Number of Manchester bit times before the command (uint).

```
aasp_device.ManchesterBitTimesBeforeCommand
```

ManchesterBitTimesAfterCommand

Number of Manchester bit times after the command (uint).

```
aasp_device.ManchesterBitTimesAfterCommand
```

ManchesterDelayAfterRead

Manchester delay after read (float).

```
aasp_device.ManchesterDelayAfterRead
```

ManchesterDelayAfterWrite

Manchester delay after write (float).

```
aasp_device.ManchesterDelayAfterWrite
```

ManchesterDelayAfterEEPROMWrite

Manchester delay after EEPROM write (float).

```
aasp_device.ManchesterDelayAfterEEPROMWrite
```

ManchesterCommandRetries

Number of Manchester command retries (uint).

```
aasp_device.ManchesterCommandRetries
```

ManchesterNeedPull-upForRead

Manchester need for pull-up for read (bool).

```
aasp_device.ManchesterNeedPull-upForRead
```

ManchesterTransmitFlag

Manchester GPIO for transmit flag (uint).

```
aasp_device.ManchesterTransmitFlag
```

ManchesterTransmitFlagActiveState

Manchester active state for transmit flag (bool).

```
aasp_device.ManchesterTransmitFlagActiveState
```

ManchesterID

Manchester ID (uint).

```
aasp_device.ManchesterID
```

ManchesterFieldSelect

Selects which half of the register is to be returned (uint).

```
aasp_device.ManchesterFieldSelect
```

SPIClockFrequency

SPI clock freqency (uint).

```
aasp_device.SPIClockFrequency
```

SPIMSBFirst

SPI bit order: When true, msb is first; otherwise, lsb is first (uint).

```
aasp_device.SPIMSBFirst
```

SPIMode

SPI mode: 0 through 3 (uint).

```
aasp_device.SPIMode
```

SPISaferSPI

If the device is using SafeSPI, the value is true (bool).

```
aasp_device.SPISaferSPI
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SPITransferSize

Width of the register, in number of bits (uint).

```
aasp_device.SPITransferSize
```

SPIWriteCommand

Bit pattern of the write command (uint).

```
aasp_device.SPIWriteCommand
```

SPIReadCommand

Bit pattern of the read command (uint).

```
aasp_device.SPIReadCommand
```

SPICommandFieldSize

Size of the command field, in bits (uint)

```
aasp_device.SPICommandFieldSize
```

SPICommandFieldShift

The command shift (uint).

```
aasp_device.SPICommandFieldShift
```

SPIAddressFieldSize

The address size, in bits (uint).

```
aasp_device.SPIAddressFieldSize
```

SPIAddressFieldShift

The address mask (uint).

```
aasp_device.SPIAddressFieldShift
```

SPIDataFieldSize

The data size, in bits (uint).

```
aasp_device.SPIDataFieldSize
```

SPIDataFieldShift

The data mask (uint).

```
aasp_device.SPIDataFieldShift
```

SPICRCFieldSize

The CRC size, in bits (uint).

```
aasp_device.SPICRCFieldSize
```

SENTNumberOfNibbles

Number of nibbles in the SENT message (uint).

```
aasp_device.SENTNumberOfNibbles
```

SENTTickTime

Tick time of the SENT message (float).

```
aasp_device.SENTTickTime
```

SENTIncludeSCNInCRC

Include the status and control nibble in the CRC of the SENT message (uint).

```
aasp_device.SENTIncludeSCNInCRC
```

SENTType

Type of SENT message (uint).

```
aasp_device.SENTType
```

SENTSensorID

Sensor identifier of the SENT message (uint).

```
aasp_device.SENTSensorID
```

SENTMaxSensorID

Maximum sensor identifier of the SENT message (uint).

```
aasp_device.SENTMaxSensorID
```

SENTTriggerWidth

SENT, TSENT: Width of the trigger (uint).

```
aasp_device.SENTTriggerWidth
```

SENTAuxPulseWidth

SENT, ASENT, SSENT, SSENT: Width of the long auxiliary pulse (uint).

```
aasp_device.SENTAuxPulseWidth
```

SENTAddressPulseWidth

SENT, ASENT: Width of the address pulse (uint).

```
aasp_device.SENTAddressPulseWidth
```

SENTSlowSerialDataMode

SENT: Slow serial data mode: 0 = standard; 1 = extended (uint).

```
aasp_device.SENTSlowSerialDataMode
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SENTSyncPulseWidth

SENT, ASENT: Width of the synchronization pulse (uint).

```
aasp_device.SENTSyncPulseWidth
```

SENTOutputPulseWidth

SENT, ASENT: Width of the output pulse (uint).

```
aasp_device.SENTOutputPulseWidth
```

SENTSamplePulseWidth

SENT, ASENT: Width of the sample pulse (uint).

```
aasp_device.SENTSamplePulseWidth
```

PulseOutput

Location where pulses are applied (uint).

```
aasp_device.PulseOutput
```

PulseOnWidth

Sets the width of the pulse-on operation (float).

```
aasp_device.PulseOnWidth
```

PulseOffWidth

Sets the width of the pulse-off operation (float).

```
aasp_device.PulseOffWidth
```

PulseVoltageLevels

Array of pulse voltages (float[]).

```
aasp_device.PulseVoltageLevels
```

PulseRisingSlewRate

Sets the rising pulse slew rate, in V/ms (float).

```
aasp_device.PulseRisingSlewRate
```

PulseFallingSlewRate

Sets the falling pulse slew rate, in V/ms (float).

```
aasp_device.PulseFallingSlewRate
```

PulseHighLevel

Sets the upper threshold level (float).

```
aasp_device.PulseHighLevel
```

PulseLowLevel

Sets the lower threshold level (float).

```
aasp_device.PulseLowLevel
```

SCClockFrequency

Clock frequency of the scan vector, in Hz.

```
aasp_device. SCClockFrequency = 10000000;
```

SVClockPolarity

Clock active polarity of the scan vector: false = active high; true = active low.

```
aasp_device.SVClockPolarity = false;
```

SVClockPhase

Clock phase of the scan vector: false = data are captured upon the rising edge and output upon the falling edge; true = data are captured upon the falling edge and output upon the rising edge.

```
aasp_device.SVClockPhase = false;
```

SVOutputPolarity

Output polarity of the scan vector: false = active low; true = active high.

```
aasp_device.SVOutputPolarity = true;
```

SVEnablePolarity

Enable polarity of the scan vector: false = active low; true = active high.

```
aasp_device.SVEnablePolarity = false;
```

SVResetBit

GPIO bit used for reset.

```
aasp_device.SVResetBit = 4;
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Device Memory Commands

These high-level commands are used to perform reads and writes of the memory of a device. Successful use of these commands requires the protocol to be set with protocol parameters that match the parameters expected by the device.

A memory address is a 32-bit unsigned number, where the lower 24 bits are the address and the upper 8 bits are the flags that describe the access. The flags are as follows:

Bit Name	Bit Number(s)	Description
WriteOnly	7	When true, the memory location is write only: it does not respond to a read
Reserved	6	Unused; reserved for future purposes
Reserved	5	Unused; reserved for future purposes
Reserved	4	Unused; reserved for future purposes
Reserved	3	Unused; reserved for future purposes
MemorySpace	2:0	0 = direct memory access 1 – 7 = indirect memory access group

Possible exceptions for all memory commands are:

- kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer.
- kCOMMUNICATIONPROTOCOLNOTSETERROR: The communication protocol is not set.
- kBADCOMMUNICATIONPROTOCOLERROR: The protocol does not support the read or write device memory.

Read(UInt32)

Reads the contents of the address from the device.

Parameter	Type	Description
address	UInt32	The address to be read
return value	UInt32	The value that was read
exception		• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed

```
uint value = aasp_device.Read(address);
```

Read(UInt32[ ])

Reads the contents of the addresses from the device.

Parameter	Type	Description
addresses	UInt32[ ]	The addresses to be read
return value	UInt32[ ]	The values that were read
exception		• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed

```
uint[] values = aasp_device.Read(addresses);
```

Read(UInt32, UInt32)

Reads the range of addresses from the device.

Parameter	Type	Description
start_address	UInt32	The address to be read
end_address	UInt32	The last address to be read
return value	UInt32[ ]	The values that were read
exception		• kCRCERROR: The protocol has a CRC, and the reply from the device failed the CRC • kREADTIMEOUTERROR: The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed

```
uint[] values = aasp_device.Read(start_address, end_
address)
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ReadField(UInt32, int, int)

Reads a field from the device.

Parameter	Type	Description
address	UInt32	The address to be read
high_bit	int	The highest bit location in the field
low_bit	int	The lowest bit location in the field
return value	UInt32	The value that was read
exception		• kCRCERROR: The protocol has a CRC and the reply from the device failed the CRC • kREADTIMEOUTERROR:The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed

```
uint value = aasp_device.ReadField(address, highBit,
lowBit);
```

ReadField(UInt32[ ], int[ ], int[ ])

Reads the fields from the device.

Parameter	Type	Description
addresses	UInt32[ ]	The addresses to be read
high_bits	int[ ]	The highest bit locations in the field
low_bits	int[ ]	The lowest bit locations in the field
return value	UInt32[ ]	The values that were read
exception		• kCRCERROR: The protocol has a CRC and the reply from the device failed the CRC • kREADTIMEOUTERROR:The device did not respond within the time required • kINDIRECTREADTIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-read operation within the time required • kGENERICREADERROR: The read operation failed

```
uint[] values = aasp_device.ReadField(addresses,
high_bits, low_bits);
```

Write(UInt32, UInt32)

Writes the value to the address on the device.

Parameter	Type	Description
address	UInt32	The address to be written
value	UInt32	The value that is to be written
exception		• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write failed

```
aasp_device.Write(address, value);
```

Write(UInt32[ ], UInt32[ ])

Writes the values to the addresses on the device.

Parameter	Type	Description
addresses	UInt32[ ]	The addresses to be written
values	UInt32[ ]	The values that are to be written
exception		• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed

```
aasp_device.Write(addresses, values);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

WriteField(UInt32, int, int, UInt32)

Writes the value to the field on the device.

Parameter	Type	Description
address	UInt32	The address to be written
high_bit	int	The highest bit location in the field
low_bit	int	The lowest bit location in the field
value	UInt32	The value that is to be written
exception		• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed

```
aasp_device.WriteField(address, high_bit, low_bit,
value);
```

WriteField(UInt32[ ], int[ ], int[ ], UInt32[ ])

Writes the values to the field on the device.

Parameter	Type	Description
addresses	UInt32[ ]	The addresses to be written
high_bits	int[ ]	The highest bit locations in the field
low_bits	int[ ]	The lowest bit locations in the field
values	UInt32[ ]	The values that are to be written
exception		Exceptions are: • kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kGENERICWRITEERROR: The write operation failed

```
aasp_device.WriteField (addresses, high_bits, low_
bits, values);
```

WriteVerify(UInt32, UInt32)

Writes the value to the address on the device, and verifies the value is correctly written.

Parameter	Type	Description
address	UInt32	The address to be written
value	UInt32	The value that is to be written
exception		• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kNOTVERIFIEDERROR: The verification operation failed • kGENERICWRITEERROR: The write operation failed

```
aasp_device.WriteVerify (address, value);
```

WriteVerify(UInt32[ ], UInt32[ ])

Writes the values to the addresses on the device, and verifies the values are correctly written.

Parameter	Type	Description
addresses	UInt32[ ]	The addresses to be written
values	UInt32[ ]	The values that are to be written
exception		• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kNOTVERIFIEDERROR: The verification operation failed • kGENERICWRITEERROR: The write operation failed

```
aasp_device.WriteVerify(addresses, values);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

WriteFieldVerify(UInt32, int, int, UInt32)

Writes the value to the field on the device, and verifies the value is correctly written.

Parameter	Type	Description
address	UInt32	The address to be written
high_bit	int	The highest bit location in the field
low_bit	int	The lowest bit location in the field
value	UInt32	The value that is to be written
exception		• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kNOTVERIFIEDERROR: The verification operation failed • kGENERICWRITEERROR: The write operation failed

```
aasp_device.WriteFieldVerify(address, high_bit, low_
bit, value);
```

WriteFieldVerify(UInt32[ ], int[ ], int[ ], UInt32[ ])

Writes the value to the field on the device, and verifies the value is correctly written.

Parameter	Type	Description
addresses	UInt32[ ]	The addresses to be written
high_bits	int[ ]	The highest bit locations in the field
low_bits	int[ ]	The lowest bit locations in the field
values	UInt32[ ]	The values that are to be written
exception		• kEEPROMWRITEDISABLEDERROR: EEPROM writes are disabled • kWRITETIMEOUTERROR:The device did not respond within the time required • kINDIRECTWRITETIMEOUTERROR: The address is an indirect address, and the device did not respond to the indirect-write operation within the time required • kNOTVERIFIEDERROR: The verification operation failed • kGENERICWRITEERROR: The write operation failed

```
aasp_device.WriteFieldVerify(addresses, high_bits,
low_bits, values);
```

Device Pulse Commands

The device pulse-sequence commands are used to send pulses on the output that is selected in the PulseOutput parameter. The PulseVoltage array is an array of voltages that is set up via the PulseVoltageLevels parameter.

SendPulseSequence(bool, int[ ])

Sends a pulse sequence, and reads the device output, if desired.

Parameter	Type	Description
read_result	bool	If true, the device is read after all the pulses have been sent
indexes	int[ ]	An array of indexes into the voltage array
return value	double	The value of the device, if requested; otherwise, 00
exception		• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kCOMMUNICATIONPROTOCOLNOTSETERROR: The communication protocol is not set • kBADCOMMUNICATIONPROTOCOLERROR: The protocol is not set to PulsesProtocol (4) • kOUTOFRANGEERROR: The pulse index is not within the PulseVoltage array bounds, or the voltage is out of range of the DAC

```
double value = aasp_device.SendPulseSequence(read_
result, indexes);
```

SendPulse(bool, double, double)

Sends a pulse, and reads the device output, if desired.

Parameter	Type	Description
read_result	bool	If true, the device is read after the pulse has been sent
voltage	double	The voltage of the pulse
width	double	The width of the pulse, in seconds
return value	double	The value of the device, if requested; otherwise, 00
exception		• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kCOMMUNICATIONPROTOCOLNOTSETERROR: The communication protocol is not set • kBADCOMMUNICATIONPROTOCOLERROR: The protocol is not set to PulsesProtocol (4) • kOUTOFRANGEERROR: The pulse index is not within the PulseVoltage array bounds, or the voltage is out of range of the DAC

```
double value = aasp_device.SendPulse(read_result,
voltage, width);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SendProgramPulses( )

Sends the program pulses.

```
aasp_device.SendProgramPulses();
```

SendPulsesReturnResponses(int, int, double)

Sends the pulses, and returns the results.

Parameter	Type	Description
count	int	The number of pulses to send
index	int	The index into the voltage array
sample_ width	double	The width between pulses where the sampling occurs, in seconds
return value	double[ ]	The values of the output of the device
exception		• kINVALIDDEVICEERROR: The device is not listed in the device table of the programmer • kCOMMUNICATIONPROTOCOLNOTSETERROR: The communication protocol is not set • kBADCOMMUNICATIONPROTOCOLERROR: The protocol is not set to PulsesProtocol (4) • kOUTOFRANGEERROR: The pulse index is not within the PulseVoltage array bounds, or the voltage is out of range of the DAC

```
public double[] SendPulsesReturnResponses(count,
pulse_value, sample_width);
```

Scan Vector Commands

Scan vectors are contained within a packed bit array that is serialized and sent to the device under test using the SPI protocol as SIN (MOSI). The SE (CS) and SCLK lines are controlled by the program. The input SO (MISO) is read and returned to the controlling PC.

Transition vectors are byte arrays that contain the state of the SE (CS), SIN (MOSI), and SCLK lines, and are clocked out by the program. There is also a reset signal. The SO (MISO) line is ignored, and a response is not returned to the PC. The bits are arranged in the SO byte as follows:

7	6	5	4	3	2	1	0
unused	unused	unused	unused	reset	SCLK	SE	SIN

ScanVector(int, byte[ ])

Performs a scan vector.

Parameter	Type	Description
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
return value	byte[ ]	The results of the scan vector; if an error occurs, a null value is possible
exception		Exception: Failure of the call operation

```
byte[] results = aasp_device.ScanVector(count,
vector);
```

LoadScanVector(UInt16, UInt16, byte[ ])

Loads a scan vector to the programmer. The scan vector identifiers start at 0 and increase to 32767.

Parameter	Type	Description
vector_id	UInt16	The vector identifier
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
exception		Exceptions: Bad vector identifier, vector too large, out of memory

```
aasp_device.LoadScanVector(vector_id, count, vector);
```

DeleteScanVector(UInt16)

Deletes a scan vector.

Parameter	Type	Description
vector_id	UInt16	The vector identifier
exception		Exception: Bad or unknown vector identifier

```
aasp_device.DeleteScanVector(vector_id);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

TransitionVector(int, byte[])

Performs a transition vector.

Parameter	Type	Description
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
exception		Exception: Failure of the call operation

```
aasp_device.TransitionVector(count, vector);
```

LoadTransitionVector(UInt16, UInt16, byte[ ])

Loads a transition vector to the programmer. The transition identifiers start at 32768 and increase to 65535.

Parameter	Type	Description
vector_id	UInt16	The vector identifier
count	int	The number of bits in the vector
vector	byte[ ]	The array of bytes containing the vector
exception		Exceptions: Bad vector identifier, vector too large, out of memory

```
aasp_device.LoadTransitionVector(vector_id, count,
vector);
```

DeleteTransitionVector(UInt16)

Deletes a transition vector.

Parameter	Type	Description
vector_id	UInt16	The vector identifier
exception		Exception: Bad or unknown vector identifier

```
aasp_device.DeleteTransitionVector(vector_id);
```

RunVectorSequence(UInt16[ ])

Runs a scan-and-transition vector sequence.

Parameter	Type	Description
sequence	int[ ]	The array of vector identifiers
exception		Exception: Bad or unknown vector identifier

```
aasp_device.RunVectorSequence(sequence);
```

Device Read Output Commands

ReadDeviceOutputVoltage( )

Reads the output voltage from the device.

Parameter	Type	Description
return value	double	The voltage from V_{OUT}, in V

```
double voltage = aasp_device.
ReadDeviceOutputVoltage();
```

ReadDeviceOutputCurrent( )

Reads the output current from the device.

Parameter	Type	Description
return value	double	The current from V_{CC}, in mA

```
double current = aasp_device.
ReadDeviceOutputCurrent();
```

ReadDeviceOutputDigital( )

Reads the output digital value from the device.

Parameter	Type	Description
return value	int	The digital value
exception		Exception: Failure of a memory-read operation

```
double value = aasp_device.ReadDeviceOutputDigital();
```

ReadDeviceOutputPWM(out double, out double)

Reads the output PWM from the device.

Parameter	Type	Description
duty_cycle	double	The duty cycle of the PWM, as a percentage
frequency	double	The frequency of the PWM, in Hz
exception		Exception: Failure of the call operation

```
aasp_device.ReadDeviceOutputPWM(out duty_cycle, out
frequency);
```

ReadDeviceOutputSENT( )

Reads the output SENT message from the device.

Parameter	Type	Description
return value	int	The SENT message with the CRC removed
exception		Exception: Timeout of the SENT read operation or failure of the CRC operation

```
uint value = aasp_device.ReadDeviceOutputSENT();
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ReadDeviceOutputSENTSlowSerialData( )

Reads the output SENT slow serial data from the device.

Parameter	Type	Description
return value	int	The SENT slow serial data with the CRC removed
exception		Exception: Timeout of the SENT read operation or failure of the CRC operation

```
uint value = aasp_device.
ReadDeviceOutputSENTSlowSerialData();
```

SendDeviceOutputSENTTrigger(double)

Sends a SENT trigger to the device.

Parameter	Type	Description
trigger_width	double	The width of the SENT trigger, in seconds
exception		Exception: Failure of the call operation

```
aasp_device.SendDeviceOutputSENTTrigger(trigger_
width);
```

Device Sample Output Commands

SampleDeviceOutputVoltage(int, double)

Samples the voltage output of the device until the specified number of samples is collected.

Parameter	Type	Description
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
exception		Exception: Failure of the call operation

```
aasp_device.SampleDeviceOutputVoltage(samples, rate);
```

SampleDeviceOutputCurrent(int, double)

Samples the current output of the device until the specified number of samples is collected.

Parameter	Type	Description
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
exception		Exception: Failure of the call operation

```
aasp_device.SampleDeviceOutputCurrent(samples, rate);
```

SampleDeviceOutputDigital(int, double)

Samples the digital output of the device until the specified number of samples is collected.

Parameter	Type	Description
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
exception		Exception: Failure of the call operation

```
aasp_device.SampleDeviceOutputDigital(samples, rate);
```

SampleDeviceOutputPWM(int, double)

Samples the duty cycle and frequency output of the device until the specified number of samples is collected.

Parameter	Type	Description
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
exception		Exception: Failure of the call operation

```
aasp_device.SampleDeviceOutputPWM(samples, rate);
```


ALLEGRO microsystems
Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

SampleDeviceOutputSENT(int, double)

Samples the SENT output of the device until the specified number of samples is collected.

Parameter	Type	Description
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
exception		Exception: Failure of the call operation

```
aasp_device.SampleDeviceOutputSENT(samples, rate);
```

SampleDeviceOutputSENTSlowSerialData(int, double)

Samples the SENT slow serial data output of the device until the specified number of samples is collected.

Parameter	Type	Description
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
exception		Exception: Failure of the call operation

```
aasp_device.SampleDeviceOutputSENTSlowSerialData(sam
ples, rate);
```

SampleDeviceMemory(int, double, UInt32[ ])

Samples the memory locations of the device until the specified number of samples is collected.

Parameter	Type	Description
samples	int	The number of samples to collect
rate	double	The rate of the sample operation, in Hz
addresses	UInt32[ ]	The addresses to sample
exception		Exception: Failure of the call operation

```
aasp_device.SampleDeviceMemory(samples, rate,
addresses);
```

ReadSampleBufferVoltages(out double[ ])

Reads the sample buffer of the device for voltage.

Parameter	Type	Description
samples	double[ ]	The samples
return value	bool	If there are more samples in the sample buffer, the value is true
exception		Exception: Failure of the call operation

```
bool more_samples = aasp_device.
ReadSampleBufferVoltages(out samples);
```

ReadSampleBufferCurrents(out double[ ])

Reads the sample buffer of the device for current.

Parameter	Type	Description
samples	double[ ]	The samples
return value	bool	If there are more samples in the sample buffer, the value is true
exception		Exception: Failure of the call operation

```
bool more_samples = aasp_device.
ReadSampleBufferCurrents (out samples);
```

ReadSampleBufferDigital(out int[ ])

Reads the sample buffer of the device for digital output.

Parameter	Type	Description
samples	int[ ]	The samples
return value	bool	If there are more samples in the sample buffer, the value is true
exception		Exception: Failure of the call operation

```
bool more_samples = aasp_device.
ReadSampleBufferDigital (out samples);
```

ReadSampleBufferPWM(out double[ ], out double[ ])

Reads the sample buffer of the device for PWM.

Parameter	Type	Description
duty_cycle_ samples	double[ ]	The array of duty cycles that were sampled
frequency_ samples	double[ ]	The array of frequencies that were sampled
return value	bool	If there are more samples in the sample buffer, the value is true
exception		Exception: Failure of the call operation

```
bool more_samples = aasp_device.
ReadSampleBufferPWM(out duty_cycle_samples, out
frequency_samples)
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

ReadSampleBufferSENT(out uint[ ])

Reads the sample buffer of the device for SENT messages.

Parameter	Type	Description
samples	uint[ ]	The samples
return value	bool	If there are more samples in the sample buffer, the value is true
exception		Exception: Failure of the call operation

```
bool more_samples = aasp_device.ReadSampleBufferSENT
(out samples);
```

ReadSampleBufferSENTSSD(out uint[ ])

Reads the sample buffer of the device for SENT slow serial data.

Parameter	Type	Description
samples	uint[ ]	The samples
return value	bool	If there are more samples in the sample buffer, the value is true
exception		Exception: Failure of the call operation

```
bool more_samples = aasp_device.
ReadSampleBufferSENTSSD (out samples);
```

ReadSampleBufferMemory(out uint[ ])

Reads the sample buffer of the device.

Parameter	Type	Description
samples	uint[ ]	The samples
return value	bool	If there are more samples in the sample buffer, the value is true
exception		Exception: Failure of the call operation

```
bool more_samples = aasp_device.
ReadSampleBufferMemory (out samples);
```

Device Serial Commands (Crocus)

ReadSerial(uint, ushort, ushort)

Reads the serial stream from the device.

Parameter	Type	Description
header	uint	Header of the read command
header bit count	ushort	Header bit count
read bit count	ushort	Number of bits to be read
return value	uint[ ]	Array of the read values
exception		• kREADTIMEOUTERROR:The device did not respond within the time required • kGENERICREADERROR: The read operation failed

WriteSerial (ushort, uint[ ])

Write the serial stream to the device.

Parameter	Type	Description
write bit count	ushort	Number of bits to be written
values	uint[ ]	Array of 32-bit words to be written
exception		• kREADTIMEOUTERROR:The device did not respond within the time required • kGENERICWRITEERROR: The write operation failed

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Bit-Field Routines

SetField(uint, uint, int, int)

Inserts the contents of a bit field into the data.

Parameter	Type	Description
data	uint	The data that contains the bit field
bitfield_value	uint	The unsigned value of the bit field to be placed in the data
stop	int	The high bit number of the field
start	int	The low bit number of the field
return value	uint	The data with the bit field inserted into it

```
uint new_data = AASP.SetField(data, bitfield_value,
stop, start);
```

SetField(uint, int, int, int)

Inserts the contents of a bit field into the data.

Parameter	Type	Description
data	uint	The data that contains the bit field
bitfield_value	int	The signed value of the bit field to be placed in the data
stop	int	The high bit number of the field
start	int	The low bit number of the field
return value	uint	The data with the bit field inserted into it

```
uint new_data = AASP.SetField(data, bitfield_value,
stop, start);
```

GetField(uint, int, int)

Extracts the contents of a bit field from the data.

Parameter	Type	Description
data	uint	The data that contains the bit field
stop	int	The high bit number of the field
start	int	The low bit number of the field
return value	uint	The unsigned bit field

```
uint bitfield = AASP.GetField(data, stop, start);
```

GetSignedField(uint, int, int)

Extracts the contents of a bit field from the data.

Parameter	Type	Description
data	uint	The data that contains the bit field
stop	int	The high bit number of the field
start	int	The low bit number of the field
return value	int	The signed bit field

```
int bitfield = AASP.GetSignedField(data, stop,
start);
```

SignExtendField(uint, int)

Sign-extends a field that is right justified.

Parameter	Type	Description
data	uint	The data that contains the bit field
width	int	The high bit number of the field
return value	int	The signed bit field

```
int bitfield = AASP.SignExtendField(data, width);
```

ConvertUnitString(double)

Converts a double to a string that represents a value that has a unit.

Parameter	Type	Description
value	double	The value
return value	string	The signed bit field

```
string val = AASP.ConvertUnitString(value);
```


Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

Revision History

Number	Date	Description
–	June 26, 2025	Initial release

Copyright 2025, Allegro MicroSystems.
Allegro MicroSystems reserves the right to make, from time to time, such departures from the detail specifications as may be required to permit improvements in the performance, reliability, or manufacturability of its products. Before placing an order, the user is cautioned to verify that the information being relied upon is current.
Allegro's products are not to be used in any devices or systems, including but not limited to life support devices or systems, in which a failure of Allegro's product can reasonably be expected to cause bodily harm.
The information included herein is believed to be accurate and reliable. However, Allegro MicroSystems assumes no responsibility for its use; nor for any infringement of patents or other rights of third parties which may result from its use.
Copies of this document are considered uncontrolled documents.

Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com